AEI Scan: a briefing for the technical team

Your team has been asked to run AEI Scan 0.1.0 in a repository where it works with a coding agent. Before you run a program on your machine, you want to know what it reads, what it sends and how it reaches its result. This briefing answers all 3, and you can check each answer against the code.

In brief:

- **What it is.** A command-line program in Python 3 that imports only the standard library. Nothing is installed.
- **What it reads.** Only the locations that each platform's vendor documents, listed in a single data file, `tool/assessment/data/platforms.json`. Section 3 is generated from that file. It opens no source file.
- **What it sends.** Nothing, unless you pass `--upload` or `--upload-file`. A single file, `tool/assessment/upload.py`, holds all its networking code. A test fails if any other module imports a networking library, and another fails if a socket is opened during a run without those options (section 4).
- **What it writes.** 3 files into a folder you name: `results.json`, `report.md` and `report.html`.
- **What the rest of this briefing holds.** How to run it, how each of the 97 checks of the Prosights Agentic Engineering Index (AEI) passes, how the score is computed, what the results file holds, and how to move a practice from Written to Checked to Enforced.

1. What you are running

The program was written and tested on Python 3.9.6 on macOS. Its tests give it the folders, environment variables and file formats that each vendor documents for Linux and Windows. On 2026-11-14 the test suite passed on GitHub's Linux and Windows machines (`ubuntu-latest`, `windows-latest`); 3 tests that need a POSIX file system are skipped on Windows. No person has run the program on a Linux or Windows workstation with real agent files.

You can read every line before you run it: `scan.py` (the scan), `answers.py` (the questions), `scoring.py` (the arithmetic), `report.py` (the outputs) and `cli.py` (the command) are in `tool/assessment/`.

2. How to run it

The commands below were run on 2026-11-14 on a made-up repository that holds the files of Claude Code and of Cursor (`tool/run_synthetic.py` builds it in a temporary folder), with that repository's path written as `REPO`. The output under each is the real output of that run. `PYTHONDONTWRITEBYTECODE=1` keeps Python from writing cache folders beside the tool.

First, see what would be read. `--dry-run` lists every location and stops:

```
PYTHONDONTWRITEBYTECODE=1 python3 assess.py REPO --dry-run --count-sessions

Platform: Claude Code
The scan would look at these locations and at nothing else:
  .claude/agents/*.md
  .claude/commands/*.md
  .claude/rules/*.md
  .claude/settings.json
  .claude/settings.local.json
  .claude/skills/*/SKILL.md
  .claude/worktrees/*
  .git/hooks/
  .github/CODEOWNERS
  .github/workflows/*.yaml
  .github/workflows/*.yml
  .mcp.json
  .worktreeinclude
  /Library/Application Support/ClaudeCode/managed-settings.d/*.json
  /Library/Application Support/ClaudeCode/managed-settings.json
  AGENTS.md
  CLAUDE.md
```

```
  CODEOWNERS
  docs/CODEOWNERS
  docs/decisions/
  ~/.claude/CLAUDE.md
  ~/.claude/projects/<project>/<session>.jsonl
  ~/.claude/projects/<project>/memory/
  ~/.claude/settings.json
Platform: Cursor
The scan would look at these locations and at nothing else:
  .agents/skills/*/SKILL.md
  .cursor/agents/*.md
  .cursor/cli.json
  .cursor/hooks.json
  .cursor/mcp.json
  .cursor/rules/*.mdc
  .cursor/skills/*/SKILL.md
  .cursor/worktrees.json
  .git/hooks/
  .github/CODEOWNERS
  .github/workflows/*.yaml
  .github/workflows/*.yml
  /Library/Application Support/Cursor/hooks.json
  AGENTS.md
  CODEOWNERS
  docs/CODEOWNERS
  docs/decisions/
  ~/.cursor/agents/*.md
  ~/.cursor/cli-config.json
  ~/.cursor/hooks.json
  ~/.cursor/mcp.json
Nothing was read and nothing was written.
```

Then run the assessment with an answers file:

```
PYTHONDONTWRITEBYTECODE=1 python3 assess.py REPO --answers answers.json \
    --count-sessions --out results

Assessed 2 coding agents in 1 run: Claude Code, Cursor. Each has its own result.
Platform: Claude Code
Result: AEI 84, Level 1 (provisional)
Overall level: AEI Level 1, Written (provisional)
Overall score: 83.8
Coverage: 53 of 97 checks (55%)
Claim gap: 5
Platform: Cursor
Result: AEI 86, Level 1 (provisional)
Overall level: AEI Level 1, Written (provisional)
Overall score: 85.9
Coverage: 52 of 97 checks (54%)
Claim gap: 0
Wrote results.json, results-cursor.json, report.md, report.html in the output folder.
    Nothing was sent anywhere.
```

The tool assesses every coding agent that has a file or folder of its own in the repository, in the same run. Each agent gets its own results file (`results.json` for the first, `results-<agent>.json` for each further one), and `report.md` and `report.html` start with 1 table of all the results. An agent that is only installed on the machine is named on the console and not assessed; `--platform` assesses it.

To answer at the keyboard in place of a file, and keep your answers for the next run:

```
python3 assess.py REPO --interactive --save-answers answers.json --out results
```

That asks 3 opening questions (the folder of your decision records, how many functions of your product you call crucial, and whether your agents run commands on another machine), the 18 survey questions and the 35 questions on what you believe is in place. Enter skips a question.

Option	What it does
`--platform NAME`	Assess this platform only: `claude-code`, `codex-cli`, `cursor`, `antigravity`, `grok-build`, `vscode-copilot` or `xcode`. Left out, every platform with a file in the repository is assessed
`--interactive`	Ask the questions at the keyboard
`--interview`	With `--interactive`: also ask the questions an assessor answers after being shown each artifact
`--answers FILE`	Read the answers from a file (section 8)
`--save-answers FILE`	With `--interactive`: keep the answers
`--count-sessions`	Count the agent's session records and read their dates. Off unless you pass it. The records are never opened
`--include-machine-identity`	Add the host name, the user name and the repository's full path to the results. Off unless you pass it
`--upload`	After writing the 3 files, send `results.json` to the results site. Off unless you pass it. The tool first prints what it will send and the address
`--upload-file FILE`	Send a `results.json` from an earlier run, and do nothing else
`--request CODE`	With `--upload` or `--upload-file`: send the results to the assessment request that gave you this code, in place of a claim code. Name 1 or more repositories; each gets its own folder under `--out`. The requester reads nothing until a person at your company agrees, signed in on the results site
`--upload-to ADDRESS`	The address to send to, in place of the built-in one. It must start with `https://`
`--benchmark`	After writing the 3 files, read the published benchmark figures from the results site and show how the result compares, on the console and in the appendix "How you compare" of the report. Off unless you pass it. The request holds no part of the result
`--benchmark-file FILE`	Compare a `results.json` from an earlier run, on the console, and do nothing else
`--benchmark-from ADDRESS`	The address of the published figures, in place of the built-in one. It must start with `https://`
`--repository-only`	Read the repository folder and nothing else: no home folder, no system folder, no environment variable
`--out FOLDER`	Where the 3 files go (default `assessment-results`)
`--dry-run`	List what would be read, then stop
`--home FOLDER`	Treat this folder as the home folder
`--version`	Print the tool's name and version, the framework's name and the schema version

With no answers the tool grades only what the scan can read and marks every question "unknown"; most dimensions then come out "not rated".

Comparing your result

`--benchmark` adds the appendix "How you compare" to the report. For the overall score and each dimension it puts your score beside 3 things:

- The results that members of the results site chose to share, for every coding agent and for yours: the median, the middle half (lower quartile to upper quartile) and where your score stands. The site gives a figure only for a group of at least 11 results; under that you see the count. These results are not a sample of all teams.
- A repository-only scan of 204 public open-source repositories that commit coding-agent files, drawn at random from a published list (286 results; the list is the supplementary material of Galster and others, arXiv 2602.14690). A full run also reads the home folder and your answers, so compare like with like: run with `--repository-only` and no answers.
- A new installation of your coding agent, with no file added: the scan passes 0 of its 35 scan checks, on every coding agent.

The tool reads the figures with 1 request that holds no part of your result. To share your own result, upload it, claim it on the results site and tick the box there; sharing is off until you do.

3. What the scan reads on your platform

Everything in this section is generated from `tool/assessment/data/platforms.json`. A note that cites "01", "05", "10" or "11" refers to the research files named under Sources. "No documented location" means none was found in the vendor's documentation read for the framework; the tool then reads nothing and the check goes to a person.

Platform	Read by the scan	Graded by a person	Checks the scan reads
Claude Code	35	0	1.4, 2.1, 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 5.2, 6.1, 6.2, 6.3, 7.2, 9.1, 9.2, 9.3, 9.4, 9.5, 9.7, 10.1, 11.1, 12.2, 13.2, 13.3, 13.4, 13.5, 13.6, 13.12, 14.1, 14.2, 14.3, 14.4, 15.11, 16.4, 16.7
Codex CLI	22	13	1.4, 2.1, 4.1, 4.2, 4.4, 6.1, 6.2, 6.3, 7.2, 9.1, 9.3, 9.4, 9.7, 10.1, 11.1, 12.2, 13.3, 13.4, 13.5, 14.2, 14.4, 16.7
Cursor	21	14	2.1, 4.1, 4.2, 4.3, 4.4, 5.2, 6.1, 6.2, 6.3, 7.2, 9.3, 9.4, 9.5, 9.7, 12.2, 13.5, 14.2, 14.3, 14.4, 16.4, 16.7
Antigravity	10	25	2.1, 4.1, 4.2, 4.3, 4.4, 6.1, 6.2, 7.2, 9.7, 14.2
Grok Build	3	32	2.1, 6.2, 7.2
VS Code with Copilot	19	16	1.4, 2.1, 4.1, 4.2, 4.3, 4.4, 6.1, 6.2, 6.3, 7.2, 9.1, 9.3, 9.4, 9.7, 12.2, 13.5, 14.2, 14.3, 16.7
Xcode	3	32	2.1, 6.2, 7.2

Check ids are explained in section 5. On Codex CLI, checks 10.1 and 11.1 are read only when the `CODEX_HOME` variable is set. The counts are for the system where the vendor documents the most: on Windows the scan reads 33 checks on Claude Code, because no page read gives the folder that holds memory and session records there, and on macOS and Windows it reads 18 on VS Code with Copilot, because the organization policy is not a file there. A location followed by a system name in parentheses is read on that system only.

Claude Code

The scan reads 35 of the 35 scan checks here; a person grades the other 0.

What	Where the scan reads	Note
Instruction file	`CLAUDE.md`, `AGENTS.md`, `~/.claude/CLAUDE.md`	
Rules scoped to paths	`.claude/rules/*.md`	with `paths` front matter

What	Where the scan reads	Note
Procedures and agent definitions	`.claude/skills/*/SKILL.md`, `.claude/agents/*.md`, `.claude/commands/*.md`	
Model pinned	`.claude/settings.json`, `.claude/agents/*.md`	an `env` name that contains MODEL, or `model` front matter, whose value holds a version number (01 section 4)
Permission rules	`.claude/settings.json`, `.claude/settings.local.json`, `~/.claude/settings.json`	the `permissions` lists
Sandbox	`.claude/settings.json`, `.claude/settings.local.json`, `~/.claude/settings.json`	key `sandbox.enabled` (01 section 13)
Network allow list	`.claude/settings.json`, `.claude/settings.local.json`, `~/.claude/settings.json`	key `sandbox.network.allowedDomains` (01 section 13)
Hooks	`.claude/settings.json`, `.claude/settings.local.json`, `~/.claude/settings.json`	key `hooks`; a plugin's `hooks/hooks.json` is not read, because its folder is not documented
Organization policy	`/Library/Application Support/ClaudeCode/managed-settings.json` (macOS), `/Library/Application Support/ClaudeCode/managed-settings.d/*.json` (macOS), `/etc/claude-code/managed-settings.json` (Linux), `/etc/claude-code/managed-settings.d/*.json` (Linux), `C:\Program Files\ClaudeCode\managed-settings.json` (Windows), `C:\Program Files\ClaudeCode\managed-settings.d\*.json` (Windows)	a policy delivered by a macOS configuration profile, the Windows registry or the vendor's server is not a file and is shown to an assessor
Memory	`~/.claude/projects/<project>/memory/` (macOS and Linux)	`<project>` is the repository's path with every character that is not a letter or a digit written as `-`, as observed on macOS on 2026-11-13; when that folder is absent the check goes to interview; on macOS and Linux only: on Linux the vendor's page shows the folder name for `/home/user/work/my-repo` as `-home-user-work-my-repo`; no page read gives the folder name on Windows, so there the check goes to interview
Session records	`~/.claude/projects/<project>/<session>.jsonl` (macOS and Linux)	counted, never opened; on macOS and Linux only: on Linux the vendor's page shows the folder name for `/home/user/work/my-repo` as `-home-user-work-my-repo`; no page read gives the folder name on Windows, so there the check goes to interview
Compaction threshold	`.claude/settings.json`, `.claude/settings.local.json`, `~/.claude/settings.json`, `.claude/agents/*.md`	key `autoCompactWindow`, or `CLAUDE_CODE_AUTO_COMPACT_WINDOW` under `env` (10 sections 1 and 2)

What	Where the scan reads	Note
Reasoning effort	`.claude/settings.json`, `.claude/settings.local.json`, `~/.claude/settings.json`, `.claude/agents/*.md`	key `effortLevel`, `CLAUDE_CODE_EFFORT_LEVEL` under `env`, or `effort` front matter
Model for agents started by an agent	`.claude/settings.json`, `.claude/settings.local.json`, `~/.claude/settings.json`, `.claude/agents/*.md`	`CLAUDE_CODE_SUBAGENT_MODEL` under `env`, or `model` front matter other than `inherit`
Prompt cache	`.claude/settings.json`, `.claude/settings.local.json`, `~/.claude/settings.json`	keys `promptCacheTtl` and `subagentPromptCacheTtl`, or `CLAUDE_CODE_PROMPT_CACHE_TTL`, `CLAUDE_CODE_SUBAGENT_PROMPT_CACHE_TTL` or `ENABLE_PROMPT_CACHING_1H` under `env`; fails when `DISABLE_PROMPT_CACHING` is set there
Usage telemetry	`~/.claude/settings.json`	`CLAUDE_CODE_ENABLE_TELEMETRY` set to 1 under `env`; the vendor reads it from user settings, managed settings or the shell, not from project settings, and the shell and managed settings are not read
MCP servers	`.mcp.json`, `.claude/settings.json`, `.claude/settings.local.json`, `~/.claude/settings.json`	the project's server file, or a permission rule that starts with `mcp__`
Agent definitions that limit tools	`.claude/agents/*.md`	`tools` or `disallowedTools` front matter
Worktrees	`.claude/worktrees/*`, `.worktreeinclude`	
Sandbox escape	`.claude/settings.json`, `.claude/settings.local.json`, `~/.claude/settings.json`	key `sandbox.allowUnsandboxedCommands` set to false (01 section 13)

Codex CLI

The scan reads 22 of the 35 scan checks here; a person grades the other 13.

What	Where the scan reads	Note
Instruction file	`AGENTS.md`	
Rules scoped to paths	No documented location	
Procedures and agent definitions	`.agents/skills/*/SKILL.md`, `.codex/agents/*.toml`	11 section 2
Model pinned	No documented location	
Permission rules	`~/.codex/config.toml`, `.codex/config.toml`	key `approval_policy`
Sandbox	`~/.codex/config.toml`, `.codex/config.toml`	key `sandbox_mode`
Network allow list	No documented location	
Hooks	`.codex/hooks.json`, `~/.codex/hooks.json`	key `hooks`, events `UserPromptSubmit`, `PreToolUse` and `Stop` (11 section 2); the inline `[hooks]` table of `config.toml` is not read

What	Where the scan reads	Note
Organization policy	`/etc/codex/requirements.toml` (Linux and macOS), `/etc/codex/managed_config.toml` (Linux and macOS), `%ProgramData%\OpenAI\Codex\requirements.toml` (Windows), `~/.codex/managed_config.toml` (Windows)	a policy delivered by the macOS preference domain `com.openai.codex` is not a file and is shown to an assessor
Memory	`$CODEX_HOME/memories/`	only when the `CODEX_HOME` variable is set; its default is not documented
Session records	`$CODEX_HOME/history.jsonl`	only when `CODEX_HOME` is set; counted, never opened
Compaction threshold	`~/.codex/config.toml`, `.codex/config.toml`	key `model_auto_compact_token_limit`
Reasoning effort	`~/.codex/config.toml`, `.codex/config.toml`, `.codex/agents/*.toml`	key `model_reasoning_effort`
Model for agents started by an agent	`~/.codex/config.toml`, `.codex/config.toml`, `.codex/agents/*.toml`	key `agents.default_subagent_model`, or `model` in an agent file
Prompt cache	No documented location	
Usage telemetry	No documented location	
MCP servers	`~/.codex/config.toml`, `.codex/config.toml`	table `mcp_servers`
Agent definitions that limit tools	No documented location	
Worktrees	`.worktreeinclude`	the worktrees themselves are under `$CODEX_HOME/worktrees`, which is not read
Sandbox escape	No documented location	

Cursor

The scan reads 21 of the 35 scan checks here; a person grades the other 14.

What	Where the scan reads	Note
Instruction file	`AGENTS.md`, `.cursor/rules/*.mdc`	
Rules scoped to paths	`.cursor/rules/*.mdc`	with `globs` front matter
Procedures and agent definitions	`.cursor/skills/*/SKILL.md`, `.agents/skills/*/SKILL.md`, `.cursor/agents/*.md`	11 section 2
Model pinned	No documented location	
Permission rules	`~/.cursor/cli-config.json`, `.cursor/cli.json`, `$CURSOR_CONFIG_DIR/cli-config.json`, `$XDG_CONFIG_HOME/cursor/cli-config.json` (Linux)	allow and deny lists
Sandbox	No documented location	
Network allow list	No documented location	

What	Where the scan reads	Note
Hooks	`.cursor/hooks.json`, `~/.cursor/hooks.json`	
Organization policy	`/Library/Application Support/Cursor/hooks.json` (macOS), `/etc/cursor/hooks.json` (Linux), `C:\ProgramData\Cursor\hooks.json` (Windows)	in WSL the Linux path applies; hooks set in the vendor's team dashboard are not a file
Memory	No documented location	
Session records	No documented location	
Compaction threshold	No documented location	
Reasoning effort	No documented location	
Model for agents started by an agent	`.cursor/agents/*.md`, `~/.cursor/agents/*.md`	`model` front matter other than `inherit`
Prompt cache	No documented location	
Usage telemetry	No documented location	
MCP servers	`.cursor/mcp.json`, `~/.cursor/mcp.json`	
Agent definitions that limit tools	`.cursor/agents/*.md`	`readonly: true` front matter
Worktrees	`.cursor/worktrees.json`	
Sandbox escape	No documented location	

Antigravity

The scan reads 10 of the 35 scan checks here; a person grades the other 25.

What	Where the scan reads	Note
Instruction file	`AGENTS.md`, `GEMINI.md`, `~/.gemini/AGENTS.md`, `~/.gemini/GEMINI.md`	
Rules scoped to paths	`.agents/rules/*.md`	05 does not state where the `glob` activation is written, so the scan checks only that a rule file exists
Procedures and agent definitions	`.agents/skills/*/SKILL.md`	11 section 2
Model pinned	No documented location	
Permission rules	No documented location	
Sandbox	No documented location	
Network allow list	No documented location	
Hooks	No documented location	
Organization policy	No documented location	
Memory	No documented location	
Session records	No documented location	
Compaction threshold	No documented location	

What	Where the scan reads	Note
Reasoning effort	No documented location	
Model for agents started by an agent	No documented location	
Prompt cache	No documented location	
Usage telemetry	No documented location	
MCP servers	`.agents/mcp_config.json`, `~/.gemini/config/mcp_config.json`	
Agent definitions that limit tools	No documented location	
Worktrees	No documented location	
Sandbox escape	No documented location	

Grok Build

The scan reads 3 of the 35 scan checks here; a person grades the other 32.

What	Where the scan reads	Note
Instruction file	No documented location	
Rules scoped to paths	No documented location	
Procedures and agent definitions	No documented location	
Model pinned	No documented location	
Permission rules	No documented location	`~/.grok/config.toml` is documented, its keys are not, so the file is not opened
Sandbox	No documented location	
Network allow list	No documented location	
Hooks	No documented location	
Organization policy	No documented location	
Memory	No documented location	
Session records	No documented location	
Compaction threshold	No documented location	
Reasoning effort	No documented location	
Model for agents started by an agent	No documented location	
Prompt cache	No documented location	
Usage telemetry	No documented location	
MCP servers	No documented location	
Agent definitions that limit tools	No documented location	
Worktrees	No documented location	
Sandbox escape	No documented location	

VS Code with Copilot

The scan reads 19 of the 35 scan checks here; a person grades the other 16.

What	Where the scan reads	Note
Instruction file	`.github/copilot-instructions.md`, `AGENTS.md`	
Rules scoped to paths	`.github/instructions/*.instructions.md`	with `applyTo` front matter
Procedures and agent definitions	`.github/skills/*/SKILL.md`, `.github/agents/*.agent.md`	11 section 2
Model pinned	No documented location	
Permission rules	`.vscode/settings.json`, `%APPDATA%\Code\User\settings.json` (Windows), `~/Library/Application Support/Code/User/settings.json` (macOS), `~/.config/Code/User/settings.json` (Linux)	keys `chat.tools.terminal.autoApprove` and `chat.tools.global.autoApprove` (https://code.visualstudio.com/docs/agents/run/approvals); settings of a profile other than the default are not read; `chat.permissions.default` is not read because the page lists no values for it
Sandbox	`.vscode/settings.json`, `%APPDATA%\Code\User\settings.json` (Windows), `~/Library/Application Support/Code/User/settings.json` (macOS), `~/.config/Code/User/settings.json` (Linux)	key `chat.agent.sandbox.enabled`, which the vendor's page says turns terminal sandboxing on or off, in Preview on macOS, Linux and WSL2 and Experimental on Windows (https://code.visualstudio.com/docs/agents/run/approvals)
Network allow list	No documented location	the keys `chat.agent.networkFilter` and `chat.agent.allowedNetworkDomains` are documented; the page read states no values for them, so the scan does not grade them
Hooks	`.github/hooks/*.json`, `~/.copilot/hooks/*.json`	key `hooks`, events `userPromptSubmitted`, `preToolUse` and `agentStop` (11 section 2)
Organization policy	`/etc/vscode/policy.json` (Linux)	on Windows the policy is in the registry under `Software\Policies\Microsoft\VSCode` and on macOS in a configuration profile; neither is a file, so it is shown to an assessor
Memory	No documented location	
Session records	No documented location	
Compaction threshold	No documented location	
Reasoning effort	No documented location	
Model for agents started by an agent	`.github/agents/*.agent.md`	`model` front matter
Prompt cache	No documented location	
Usage telemetry	No documented location	
MCP servers	`.mcp.json`	
Agent definitions that limit tools	`.github/agents/*.agent.md`	`tools` front matter

What	Where the scan reads	Note
Worktrees	No documented location	
Sandbox escape	No documented location	

Xcode

The scan reads 3 of the 35 scan checks here; a person grades the other 32.

What	Where the scan reads	Note
Instruction file	No documented location	the folders `~/Library/Developer/Xcode/CodingAssistant/ClaudeAgentConfig`, `.../codex` and `.../gemini` are documented, the file names in them are not
Rules scoped to paths	No documented location	
Procedures and agent definitions	No documented location	
Model pinned	No documented location	
Permission rules	No documented location	Settings, Intelligence, Permissions; nothing on disk is documented
Sandbox	No documented location	
Network allow list	No documented location	
Hooks	No documented location	
Organization policy	No documented location	the MDM key `CodingAssistantAllowExternalIntegrations` is documented; it is not a file
Memory	No documented location	
Session records	No documented location	
Compaction threshold	No documented location	
Reasoning effort	No documented location	
Model for agents started by an agent	No documented location	
Prompt cache	No documented location	
Usage telemetry	No documented location	
MCP servers	No documented location	
Agent definitions that limit tools	No documented location	
Worktrees	No documented location	
Sandbox escape	No documented location	

The repository itself, on every platform

- `.git/hooks/`
- `.github/workflows/*.yml`
- `.github/workflows/*.yaml`
- `.github/CODEOWNERS`
- `CODEOWNERS`
- `docs/CODEOWNERS`
- The folder of decision records you name, read only when it is inside the repository.

Linux and Windows

Platform	Linux	Windows	Vendor pages read
Claude Code	User files in `~/.claude`, or in the folder `CLAUDE_CONFIG_DIR` names. Managed settings in `/etc/claude-code/` (also in WSL). Memory and session records in `~/.claude/projects/<project>/`.	User files in `%USERPROFILE%\.claude`, or in the folder `CLAUDE_CONFIG_DIR` names. Managed settings in `C:\Program Files\ClaudeCode\`. A registry policy is not read. No page read gives the name of the project folder, so memory and session records go to a person.	https://code.claude.com/docs/en/settings, https://code.claude.com/docs/en/managed-settings.md, https://code.claude.com/docs/en/claude-directory.md
Codex CLI	User files in `~/.codex`, or in the folder `CODEX_HOME` names. Requirements and managed configuration in `/etc/codex/`.	User files in `~/.codex`, or in the folder `CODEX_HOME` names. Requirements in `%ProgramData%\OpenAI\Codex\requirements.toml`; managed configuration in `~/.codex/managed_config.toml`.	https://learn.chatgpt.com/docs/config-file/config-reference, https://learn.chatgpt.com/docs/enterprise/managed-configuration
Cursor	User files in `~/.cursor`; the CLI configuration also in `$CURSOR_CONFIG_DIR` or `$XDG_CONFIG_HOME/cursor/`. Enterprise hooks in `/etc/cursor/hooks.json` (also in WSL).	User files in `%USERPROFILE%\.cursor`; the CLI configuration also in `$CURSOR_CONFIG_DIR`. Enterprise hooks in `C:\ProgramData\Cursor\hooks.json`.	https://cursor.com/docs/agent/hooks, https://cursor.com/docs/cli/reference/configuration
Antigravity	The vendor's page writes every user location as `~/.gemini/...` and names no system. The scan reads the same locations under the home folder.	The vendor's page writes every user location as `~/.gemini/...` and names no Windows path. The scan reads the same locations under the home folder.	https://antigravity.google/docs/rules
Grok Build	`~/.grok/config.toml` is documented; its keys are not, so the file is not opened.	`%USERPROFILE%\.grok\config.toml` is documented; its keys are not, so the file is not opened.	https://docs.x.ai/build/overview
VS Code with Copilot	User settings in `~/.config/Code/User/settings.json`. Policy in `/etc/vscode/policy.json`. User hooks in `~/.copilot/hooks/`, or in the folder `COPILOT_HOME` names.	User settings in `%APPDATA%\Code\User\settings.json`. Policy is in the registry and is not read. User hooks in `~/.copilot/hooks/`, or in the folder `COPILOT_HOME` names.	https://code.visualstudio.com/docs/configure/settings, https://code.visualstudio.com/docs/enterprise/policies, https://code.visualstudio.com/docs/copilot/customization/hooks, https://code.visualstudio.com/docs/languages/json

Platform	Linux	Windows	Vendor pages read
Xcode	Xcode is a macOS application; nothing is read.	Xcode is a macOS application; nothing is read.	none

- ~ is your home folder; on Windows that is `%USERPROFILE%`. A location written with `%APPDATA%`, `%ProgramData%`, `$XDG_CONFIG_HOME` or `$CURSOR_CONFIG_DIR` is read only when the variable is set. No vendor location read is under `%LOCALAPPDATA%`. `CLAUDE_CONFIG_DIR`, `CODEX_HOME` and `COPILOT_HOME`, when set, replace `~/.claude`, `~/.codex` and `~/.copilot`.
- An organization policy that is not a file is never read: a macOS configuration profile, the Windows registry, a vendor's server or dashboard.
- A UTF-8 byte-order mark and CRLF line endings are read. VS Code's `settings.json` may hold comments; every other JSON file must be plain JSON.
- On macOS and Linux a git hook counts only when it is executable, because git ignores one that is not (https://git-scm.com/docs/githooks). Windows has no executable bit, so there the file's presence counts.
- Inside WSL the tool is on Linux. Run from Windows on a repository reached as `\\wsl$\...`, it reads the repository only and tells you to run it inside WSL.
- Xcode runs on macOS only; on Linux and Windows the tool does not look for it.
- Every Linux and Windows behavior is tested by simulation on macOS (`tool/tests/test_systems.py`): the tool is given the other system's name, variables and folder layout. The suite also runs on GitHub's `ubuntu-latest` and `windows-latest` machines; section 4 gives the last result.

What the scan does in those locations:

- It reads whether a file exists, and its name when it is inside the repository.
- In agent settings files it reads the keys the checks name: whether the sandbox is on and closed, whether a network list, permission rules and hooks exist, which events and tools the hooks are registered on and whether each refuses when it fails, and whether a compaction threshold, a reasoning effort, a model for agents started by an agent, a cache lifetime and usage telemetry are set. It never reads a hook's command and never copies a value.
- It reads whether an MCP server file holds any entry, and whether a worktree folder or setup file exists.
- It reads the front matter of rule files and agent definitions.
- It reads the text of instruction files to find headings, command lines and the words "never", "must not" and "do not", and counts their lines. It keeps only the result.
- It reads the names of the files in `.git/hooks/` and `.github/workflows/`, and on macOS and Linux whether each git hook file is executable.
- It counts the files in the agent's memory folder. With `--count-sessions` it counts session records and reads their dates.

4. The privacy statements, and the test behind each

The tests are in `tool/tests/`. Run them yourself with the command at the end of this section.

Statement	Test
The tool imports only the Python standard library	`test_privacy.py`: `test_only_standard_library_imports`
Only `upload.py` imports a networking library, and no module imports a process library	`test_privacy.py`: `test_no_networking_or_process_imports`
Without `--upload` and `--benchmark`, no socket is opened during a run, on any of the 7 platforms, and nothing is sent	`test_privacy.py`: `test_no_socket_is_opened_during_a_run`; `test_upload.py`: `test_without_the_flag_nothing_is_sent`

Statement	Test
With `--upload`, the tool sends `results.json` and nothing else, after printing its size, the address and what it holds	`test_upload.py`: `test_upload_sends_the_results_file_and_nothing_else`
With `--request`, the tool sends each repository's results file with the request code and the name of the repository's folder, after printing that name, and without `--upload` it sends nothing	`test_upload.py`: `test_request_sends_each_repository_with_the_code_and_its_name`, `test_request_without_upload_sends_nothing`
The tool says before sending when the machine's identity is in the file	`test_upload.py`: `test_machine_identity_is_named_before_it_is_sent`
The tool sends only to an `https://` address and follows no redirect	`test_upload.py`: `test_no_address_and_a_plain_http_address_send_nothing`, `test_a_redirect_is_not_followed`
With `--benchmark`, the tool makes 1 GET request with no body and no cookie, to the address it printed first, and prints nothing the site sends except numbers	`test_benchmark.py`: `test_benchmark_reads_the_figures_and_sends_no_part_of_the_result`, `test_nothing_the_site_sends_is_printed_except_numbers`
With `--repository-only`, no file outside the repository is opened	`test_benchmark.py`: `test_nothing_outside_the_repository_is_read`
The tool prints the SHA-256 of the file before it sends and warns when the site reports another value	`test_upload.py`: `test_upload_sends_the_results_file_and_nothing_else`, `test_another_digest_from_the_site_is_said`
With files of 2 coding agents in the repository, 1 run writes 1 result for each and 1 report for both, and an upload sends 1 file for each	`test_combined.py`: `test_two_agents_found_give_two_results_and_one_report`, `test_upload_sends_one_file_per_agent`
An agent that is only installed on the machine is named and not assessed	`test_combined.py`: `test_an_agent_only_installed_on_the_machine_is_named_not_assessed`
Section 7 of the report names only files, keys and hook events that the vendor documents, and names no file where the vendor documents none	`test_combined.py`: `test_open_checks_name_the_vendors_file_and_key`, `test_no_documented_location_names_no_file`, `test_every_file_and_key_named_comes_from_the_platform_data`
No source file, memory file or session record is opened	`test_privacy.py`: `test_source_files_are_never_opened`
A secret planted in a settings file, and planted lines of source and instruction text, appear in no output	`test_privacy.py`: `test_planted_secret_and_source_text_never_in_outputs`
No user name, host name or home path appears in any output unless you ask	`test_privacy.py`: `test_no_user_host_or_home_path_in_outputs`, `test_machine_identity_only_on_opt_in`
Session records are untouched without `--count-sessions`, and only counted with it	`test_scan.py`: `test_sessions_not_read_without_opt_in`, `test_sessions_counted_with_opt_in`
A folder you name is read only if it is inside the repository	`test_scan.py`: `test_named_folder_outside_repository_is_not_read`
`--dry-run` lists the locations and reads and writes nothing	`test_cli.py`: `test_dry_run_lists_paths_and_writes_nothing`
Where the vendor documents no location, the check goes to a person	`test_scan.py`: `test_no_documented_location_becomes_interview_item`
The HTML report loads no outside script, style, font or image	`test_cli.py`: `test_html_report_is_self_contained`

```
cd tool
PYTHONDONTWRITEBYTECODE=1 python3 -m unittest discover -s tests -t .
```

On 2026-11-16 this ran 217 tests on macOS and all passed. On 2026-11-14 the suite of 216 tests passed on GitHub's `ubuntu-latest` and `windows-latest` machines (GitHub Actions run 43648860176). 3 tests are skipped on Windows because they need a file system with an executable bit and with `\` allowed in a file name: the git hook without the executable bit, the folder named with backslashes on Linux, and the WSL command line. 66 of the 217, in `test_systems.py`, give the tool the name, variables and folder layout of Linux or Windows. The tests build their repositories in temporary folders. 3 of them compare the tool's data with the framework's tables, so they need `06-framework.md` 1 folder above `tool`.

5. The 97 checks

Generated from `tool/assessment/data/catalog.json`. The level of a check is also its weight in the score. The column "What the check reads" says in words what the scan, the survey or the assessor looks at; section 3 gives the location on each platform. In `results.json` the field `evidence` holds the catalog's code for a scan check, a word in capitals that names a row of the tables in section 3: INSTR is the instruction file, SCOPED the rules scoped to paths, PROC the procedures and agent definitions, PERM the permission rules, NET the network allow list, MANAGED the organization policy, COMPACT the compaction threshold, EFFORT the reasoning effort, SUBMODEL the model for agents started by an agent, CACHE the prompt cache, TELEMETRY the usage telemetry, MCP the MCP servers, AGENTDEF the agent definitions that limit tools, WORKTREE the worktrees, ESCAPE the sandbox escape setting, REPO the repository itself.

1. Requirements and recording

Check	Level	Method	What the check reads	Passes when
1.1	1	Survey	Your answer to the survey question "Are work items written down before an agent starts on them?"	The client names the place
1.2	2	Interview	The check that refuses a change tied to no item	The check is shown and refuses a test change
1.3	2	Interview	A list of open work that a script regenerates	The list and the script that writes it are shown
1.4	3	Scan	A hook on the prompt-submit event (Claude Code and Codex CLI `UserPromptSubmit`; VS Code `userPromptSubmitted`; on other platforms, shown to a person)	1 such hook is registered

2. Decisions

Check	Level	Method	What the check reads	Passes when
2.1	1	Scan	The decision-record folder you name	The folder holds at least 1 record
2.2	2	Interview	The check that reads decision records at commit	The check is shown
2.3	1	Survey	Your answer to the survey question "Is it written which choices a person makes and which an agent may make?"	Yes, with the document named

3. Expected behavior and design

Check	Level	Method	What the check reads	Passes when
3.1	1	Interview	For 1 recent change: the expected behaviors written before the build	The list is shown and predates the change
3.2	2	Interview	The check that ties each expected behavior to a test	The check is shown

4. Agent briefing and model policy

Check	Level	Method	What the check reads	Passes when
4.1	1	Scan	The agent's instruction files	At least 1 instruction file exists
4.2	1	Scan	Headings and command lines of the instruction files	At least 4 of GitHub's 6 areas appear (commands, testing, project structure, code style, git workflow, boundaries)
4.3	1	Scan	Rule files that apply only to some paths	At least 1 path-scoped rule file exists
4.4	1	Scan	Procedure files and agent definitions	At least 1 procedure or agent definition exists
4.5	3	Scan	The model named in settings and in agent definitions	A model is named with a version number, not by an alias
4.6	3	Scan	A hook on the tool that starts another agent (Claude Code `PreToolUse` on Agent; on other platforms, shown to a person)	1 such hook is registered

5. Build

Check	Level	Method	What the check reads	Passes when
5.1	1	Survey	Your answer to the survey question "Does each agent work in its own checkout or environment?"	Yes, with the mechanism named
5.2	3	Scan	A hook before shell commands (Claude Code `PreToolUse` on Bash; Cursor `beforeShellExecution`; on other platforms, shown to a person)	1 such hook is registered
5.3	2	Interview	The limit that stops a build or test run that hangs	The limit is shown
5.4	1	Survey	Your answer to the survey question "Is there a written limit on the size of 1 agent task?"	Yes, with the limit

6. Test and evidence

Check	Level	Method	What the check reads	Passes when
6.1	1	Scan	Command lines of the instruction files	A command line or a heading names tests
6.2	2	Scan	A file in `.git/hooks/` that is not a sample, or a workflow file in `.github/workflows/`	At least 1 exists
6.3	3	Scan	A hook on the end of the agent's turn (Claude Code and Codex CLI `Stop`; Cursor `stop`; VS Code `agentStop`; on other platforms, shown to a person)	1 such hook is registered
6.4	4	Interview	Test results kept as files for each build	Results of the last 2 builds are shown

Check	Level	Method	What the check reads	Passes when
6.5	4	Interview	For each function you name as crucial: a test set run at every release	The release step that runs each set is shown
6.6	2	Interview	The gate that fails on new warnings in changed files	The gate is shown

7. Review and merge

Check	Level	Method	What the check reads	Passes when
7.1	1	Survey	Your answer to the survey question "Is every agent change reviewed before it merges by a reviewer other than its author: a person, or a model run as a separate reviewer?"	Yes
7.2	2	Scan	A `CODEOWNERS` file; when none is found, branch protection shown to a person	Either exists
7.3	2	Interview	An automated review that runs on every change before it merges and cannot be skipped	The mechanism is shown
7.4	2	Interview	The merge step that builds and checks after each merge and stops on failure	The step is shown
7.5	1	Survey	Your answer to the survey question "Is it written who may start or assign an agent?"	Yes
7.6	3	Interview	The written rule that says which changes need a person's approval before they merge or ship, by risk	The rule is shown
7.7	4	Interview	The record of automated approvals, and the sample of them a person reviewed in the last quarter	Both are shown

8. Release

Check	Level	Method	What the check reads	Passes when
8.1	1	Survey	Your answer to the survey question "Does a person confirm each release?"	Yes
8.2	2	Interview	The release gate that refuses when a required item is missing	The gate is shown
8.3	4	Interview	The record of each release's gate result	Records of the last 2 releases are shown

9. Safety controls

Check	Level	Method	What the check reads	Passes when
9.1	3	Scan	The sandbox setting	The sandbox is on; Codex `sandbox_mode` is not `danger-full-access`
9.2	3	Scan	The setting that limits network access	Network access is limited to a list

Check	Level	Method	What the check reads	Passes when
9.3	3	Scan	The permission rules in the agent's settings	Permission rules exist and no setting approves everything (Codex `approval_policy` is not `never`; VS Code `chat.tools.global.autoApprove` is off)
9.4	3	Scan	A hook that can deny a tool call (Claude Code and Codex CLI `PreToolUse`; Cursor `preToolUse`, `beforeShellExecution`, `beforeMCPExecution`, `beforeReadFile`; VS Code `preToolUse`; on other platforms, shown to a person)	1 such hook is registered
9.5	3	Scan	What happens when the hook itself fails	The action is refused; on Claude Code, every `PreToolUse` hook has `onFailure` set to `block`; on Cursor, every hook entry has `failClosed: true`
9.6	2	Interview	The secret scan that runs before code leaves the machine	The scan and its refusal are shown
9.7	1	Scan	Lines of the instruction files that hold "never", "must not" or "do not"	At least 1 such line exists
9.8	4	Interview	A list of the team's rules marked enforced, checked or written only	The list is shown and is less than 6 months old

10. Memory and continuity

Check	Level	Method	What the check reads	Passes when
10.1	1	Scan	How many files the agent's memory folder holds, never their contents	At least 1 file
10.2	2	Interview	The end-of-session step that refuses to finish with failing tests or unpushed work	The step is shown
10.3	1	Survey	Your answer to the survey question "Is the start-of-session procedure written?"	Yes
10.4	1	Survey	Your answer to the survey question "Does each agent write down what it learned from its work, in a place the next agent reads?"	Yes
10.5	2	Interview	The mechanism that refuses a change whose agent recorded no lesson entry	The mechanism is shown
10.6	2	Interview	For the last 5 agent changes, the record that the agent read the carried-forward lessons before its work	The record is shown for all 5

11. Measurement

Check	Level	Method	What the check reads	Passes when
11.1	4	Scan	The count and dates of the agent's session records, never their text, and only when you turn the count on	At least 1 record is dated in the last 30 days
11.2	4	Interview	Merge rate and revert rate of agent changes against the team's own baseline	Both numbers for the last quarter are shown
11.3	4	Interview	Usage or cost recorded per agent or task	1 report is shown
11.4	4	Interview	A quality trend: warnings, complexity or failures over time	1 trend covering 3 months is shown
11.5	1	Survey	Your answer to the survey question "Do you measure agent work by a named outcome?"	1 measure is named

12. Governance

Check	Level	Method	What the check reads	Passes when
12.1	1	Survey	Your answer to the survey question "Is there a written statement of which AI tools are used, on which data, and who approves?"	Yes, with the document named
12.2	3	Scan	The organization's policy file for the agent	An organization-level policy is in force on developers' machines
12.3	1	Interview	Rules with stable identifiers and a record of why each exists	The rules and the record are shown
12.4	1	Survey	Your answer to the survey question "Is agent use covered by your organization's AI risk process?"	Yes

13. Token and cost

Check	Level	Method	What the check reads	Passes when
13.1	1	Survey	Your answer to the survey question "Is it written which model and which reasoning effort each kind of task uses?"	Yes, with the place named
13.2	1	Scan	The number of lines in each instruction file, never the lines	Every instruction file found has fewer than 200 lines
13.3	3	Scan	The setting for compacting the conversation	A threshold at which the conversation is compacted is set
13.4	3	Scan	The reasoning-effort setting	A reasoning effort is set in settings or in an agent definition
13.5	3	Scan	The model set for agents that another agent starts	Agents that another agent starts are given a model in settings or in an agent definition
13.6	3	Scan	The prompt-cache settings	A prompt cache lifetime is set and caching is not turned off
13.7	4	Interview	A report that separates input tokens read from the cache from input tokens sent uncached	1 report is shown
13.8	2	Interview	A mechanism that shortens long command output before the agent reads it	The mechanism is shown

Check	Level	Method	What the check reads	Passes when
13.9	1	Survey	Your answer to the survey question "Is it written that long or verbose work goes to its own agent, and that agents wait without polling?"	Yes
13.10	3	Interview	A spend or usage limit set at the provider, for the workspace or for each person	The setting is shown
13.11	4	Interview	A review of tokens or cost per model over time, and 1 change made because of it	The review and the change are shown
13.12	4	Scan	The usage-telemetry setting	Usage telemetry is turned on

14. Harness depth

Check	Level	Method	What the check reads	Passes when
14.1	3	Scan	A hook that runs before the agent writes or edits a file (Claude Code `PreToolUse` on Write or Edit)	1 such hook is registered
14.2	1	Scan	The MCP server declarations and the permission rules	MCP servers are declared in a file, or a permission rule names an MCP tool
14.3	3	Scan	The tool limits in agent definitions	At least 1 agent definition limits the tools its agent may use
14.4	1	Scan	Worktree folders and the worktree setup file	A worktree folder the tool made, or a worktree setup file, exists
14.5	2	Interview	The command that shows the state of each background or unattended run	The command and its output are shown
14.6	3	Interview	The channel through which an agent runs a command on another machine	The channel runs only commands on an allow list, and the list is shown
14.7	2	Interview	The tests of the team's own hooks and guard scripts	The tests are shown

15. Process guards

Check	Level	Method	What the check reads	Passes when
15.1	3	Interview	The script every delete goes through, and the hook that refuses a delete made any other way	Both are shown
15.2	3	Interview	The hook that refuses a build or test command run outside the team's wrapper script	The hook is shown
15.3	2	Interview	The check that keeps credentials and real messages out of test fixtures	The check and its refusal are shown
15.4	1	Survey	Your answer to the survey question "Is it written that a new integration or tool starts with no access until it is listed?"	Yes
15.5	2	Interview	What stops a test from reaching a live account unless a person turns that on	The mechanism is shown
15.6	2	Interview	The commit check that refuses personal identifiers and authorship marks	The check is shown
15.7	2	Interview	The set of injection attempts run against a model before it is given a channel, and the step that runs it	Both are shown

Check	Level	Method	What the check reads	Passes when
15.8	1	Survey	Your answer to the survey question "Is it written that a change touches only what its own work item needs?"	Yes
15.9	2	Interview	For 1 guard: its single switch, and the test of the off state	Both are shown
15.10	3	Interview	The script that removes an agent's workspace and refuses when its work is not merged	The script and its refusal are shown
15.11	3	Scan	The setting that lets a command run outside the sandbox	No command can leave the sandbox: Claude Code `sandbox.allowUnsandboxedCommands` is false

16. Model conduct

Check	Level	Method	What the check reads	Passes when
16.1	1	Survey	Your answer to the survey question "Is there a written record of how each model you use has failed in your own work?"	Yes
16.2	1	Survey	Your answer to the survey question "Does every brief to an agent state what done means?"	Yes
16.3	2	Interview	For the last 2 agent reports: who other than the agent checked the claims, and what they checked	Both are shown
16.4	2	Scan	A hook on a denied tool call (Claude Code `PermissionDenied`; Cursor `postToolUseFailure`; on other platforms, shown to a person)	1 such hook is registered
16.5	2	Interview	The log of refused calls, each with the model that made the call	The log is shown, with a model on each line
16.6	3	Interview	The mechanism that refuses the end of a turn while tasks are open, and 1 refusal it made	Both are shown
16.7	3	Scan	A hook on the end of a started agent's work or on a task being marked complete (Claude Code `SubagentStop`, `TaskCompleted`; Codex CLI `SubagentStop`; Cursor `subagentStop`; VS Code `subagentStop`; on other platforms, shown to a person)	1 such hook is registered
16.8	4	Interview	Counts per model for 2 periods (rule breaks, early stops, false "done" reports) and the schedule on which they are read	The counts for 2 periods and the schedule are shown
16.9	4	Interview	The measurement made before the last new model version was adopted, and the line of the model policy it changed or confirmed	Both are shown

6. How your answers and the scan combine

Each check ends with 1 of 6 results.

Result	When	Credit	Counts as passed for a level
Pass	The evidence meets the criterion	1	Yes

Result	When	Credit	Counts as passed for a level
Claimed	A survey "yes" on a check of level 2 or higher, or an interview answer with no artifact shown	0.5	No
Fail	The evidence was read and does not meet the criterion, or the answer is "no"	0	No
Declined	You would not answer or would not allow the read	0	No
Unknown	It could not be established for a reason that is not your choice: a question not asked, a settings file that could not be parsed	Left out	Left out
Not applicable	The practice cannot exist for you. Only 2 checks can be: 6.5, when you name no crucial function, and 14.6, when no agent of yours runs commands on another machine	Left out	Left out

For each of the 35 scan checks the tool also takes what you believe is in place (yes, no, do not know). Then:

1. Where the scan can read the evidence, the scan result stands.
2. You said yes and the scan found nothing: Fail, listed under "Said but not found". The count of these is the claim gap.
3. You said no and the scan found it: Pass, listed under "In place but not known to the team".
4. Your answer never raises a scan check above Fail.

A scan check the tool cannot read for you (no documented location on your platform, session counting not turned on, a named folder outside the repository, no `CODEOWNERS` file) is graded by a person: by the interview answer when there is one; otherwise by your belief, where "yes" is a Pass at level 1 and Claimed above it and "no" is a Fail; otherwise it is Unknown.

7. The arithmetic

For 1 dimension:

- Weight of a check = its level, 1 to 4.
- Coverage = graded checks ÷ applicable checks. Graded means Pass, Claimed, Fail or Declined. Applicable means all but Not applicable.
- Below 50% coverage the dimension is not rated.
- Score = 100 × (sum of weight × credit over graded checks) ÷ (sum of weight over graded checks), rounded to a whole number, halves up.
- Level = the highest level L that has at least 1 graded check, such that at L and at every lower level with a graded check, at least 80% of the graded checks are Pass. A level with no graded check neither raises nor holds the level. If the lowest level with a graded check fails, the level is 0.
- Cap = the highest level at which the dimension has any check: 2 for dimensions 2, 3, 7 and 10; 3 for dimensions 1, 4, 5, 12, 14 and 15; 4 for dimensions 6, 8, 9, 11, 13 and 16.

Overall:

- Score = the mean of the rated dimensions' scores before rounding, to 1 decimal place.
- Level = the highest L from 1 to 4 for which all of these hold: at least 75% of the rated dimensions have a level of at least min(L, that dimension's cap); dimension 9 is rated and its level is at least min(L, 3); and for L = 4, dimension 11 is rated and at level 4.
- The level is provisional when any dimension is not rated. When no L holds, the level is 0. When dimension 9 is not rated, no level is given.

Worked example, dimension 9, with these results: 9.1 Pass, 9.2 Pass, 9.3 Pass, 9.4 Pass, 9.5 Fail, 9.6 Pass, 9.7 Pass, 9.8 Pass.

- Weights: 3, 3, 3, 3, 3, 2, 1, 4. Sum over graded checks = 22.

- Weighted credit = 3 + 3 + 3 + 3 + 0 + 2 + 1 + 4 = 19.
- Score = 100 × 19 ÷ 22 = 86. Coverage = 8 ÷ 8 = 100%.
- Level 1: 1 of 1 passed (9.7). Level 2: 1 of 1 (9.6). Level 3: 4 of 5 = 80%, which meets the threshold. Level 4: 1 of 1 (9.8). The dimension is at level 4.

If 9.6 had been a "yes" with nothing shown, it would be Claimed: credit 0.5, score = 100 × 18 ÷ 22 = 82, and level 2 would have 0 of 1 passed, so the dimension would stop at level 1.

The tests `test_5_5_worked_example` and `test_5_5_worked_example_variant` in `tool/tests/test_scoring.py` compute both cases.

The run on the repository of Kyberna on 2026-11-16, with Claude Code, gave these 16 dimensions, and the result "AEI 79, Level 1":

#	Dimension	Level	Score	Checks graded
1	Requirements and recording	3, Enforced	100	4 of 4
2	Decisions	2, Checked	100	3 of 3
3	Expected behavior and design	2, Checked	100	2 of 2
4	Agent briefing and model policy	3, Enforced	100	6 of 6
5	Build	3, Enforced	100	4 of 4
6	Test and evidence	2, Checked	56	6 of 6
7	Review and merge	1, Written	47	7 of 7
8	Release	4, Measured	100	3 of 3
9	Safety controls	4, Measured	86	8 of 8
10	Memory and continuity	1, Written	78	6 of 6
11	Measurement	1, Written	53	5 of 5
12	Governance	1, Written	25	2 of 4
13	Token and cost	3, Enforced	88	12 of 12
14	Harness depth	3, Enforced	100	7 of 7
15	Process guards	3, Enforced	100	11 of 11
16	Model conduct	1, Written	27	9 of 9

8. The results file and the answers file

This section is `tool/SCHEMA.md` from its "Top level" heading on, included as written. Schema version: "1.0".

Top level

Field	Type	Meaning
`schema_version`	string	"1.0"
`tool`	object	`name`, `version` of the tool that wrote the file
`framework`	object	`name` of the framework and `catalog_version`, the date of the check catalog
`generated`	string	UTC time the file was written, `YYYY-MM-DDTHH:MM:SSZ`

Field	Type	Meaning
`platform`	string	The coding agent assessed: `claude-code`, `codex-cli`, `cursor`, `antigravity`, `grok-build`, `vscode-copilot` or `xcode`
`also_detected`	list of strings	Other platforms whose markers were found. One with a file in the repository has its own results file from the same run, `results-<platform>.json`, in this format; one found only on the machine was not assessed
`options`	object	`count_sessions`, `machine_identity`, `answers`: true or false for each choice the client made
`overall`	object	See below
`dimensions`	list of 16 objects	See below
`checks`	list of 97 objects	See below
`said_but_not_found`	list of check ids	The answer was yes and the scan found nothing
`in_place_but_not_known`	list of check ids	The answer was no and the scan found it
`next_actions`	list of at most 10 objects	See below, in the order to act
`read`	object	`paths`: every location the scan looked at. `questions`: the ids answered, in 3 lists, `survey`, `beliefs`, `interview`
`model_conduct`	list of at most 12 objects	The counts the client gave for each model, copied from the answers file; empty when none were given. The tool counts none of them and they change no grade. See below
`limits`	list of 9 strings	What the result does not show
`machine`	object	Present only when the client asked for it: `host`, `user`, `repository_path`

`overall`

Field	Type	Meaning
`score`	number or null	Mean of the rated dimensions' scores, 0 to 100, 1 decimal place; null when no dimension is rated
`level`	0 to 4, or null	null when dimension 9 could not be rated
`level_name`	string	Unrecorded, Written, Checked, Enforced, Measured, or "not rated"
`provisional`	true or false	true when at least 1 dimension is not rated
`coverage`	object	`graded`, `applicable`, `percent` over all checks
`claim_gap`	number	The length of `said_but_not_found`

Each of `dimensions`

Field	Type	Meaning
`number`	1 to 16	
`name`	string	
`rated`	true or false	false when fewer than half of the applicable checks could be graded
`score`	0 to 100, or null	Whole number; null when not rated
`level`	0 to 4, or null	null when not rated
`level_name`	string	As in `overall`
`cap`	2, 3 or 4	The highest level this dimension has a check for
`coverage`	object	`graded`, `applicable`, `percent`
`meaning`	string	1 sentence on what the level means

Each of `checks`

Field	Type	Meaning
`id`	string	"1.1" to "16.9"
`dimension`	1 to 16	
`level`	1 to 4	Also the check's weight
`method`	string	How the framework grades it: `scan`, `survey` or `interview`
`graded_by`	string	What graded it in this run: `scan`, `survey` or `interview`. A scan check with no documented location on the platform is graded by `interview` or `survey`
`result`	string	`pass`, `claimed`, `fail`, `declined`, `unknown` or `not_applicable`
`disagreement`	string or null	`said_but_not_found`, `in_place_but_not_known`, or null
`reason`	string	Why the result is what it is, in words; never a value copied from a file
`evidence`	string	What the check reads, from the catalog
`locations`	list of strings	Where the scan looked for it on this platform; empty for a check a person grades
`found`	list of strings	What the scan found there: at most 5 names and a count of the rest
`pass`	string	The pass criterion, from the catalog
`source`	string	The framework's source for the check

Each of `next_actions`

Field	Type	Meaning
`check`	string	The check id
`dimension`	1 to 16	
`level`	1 to 4	
`result`	string	`fail`, `claimed` or `declined`
`action`	string	The evidence that would pass the check

Field	Type	Meaning
`where`	list of strings	The locations on this platform where that evidence goes; empty when a person must be shown it

Each of `model_conduct`

Field	Type	Meaning
`model`	string of at most 60 characters	The model's name as the client writes it
`period`	string of at most 40 characters	The period counted, for example "2026-10"
`rule_breaks`	whole number, 0 to 1,000,000	Times the model broke a written rule
`early_stops`	whole number	Times it stopped before the work was finished, or asked for a permission the instructions already gave
`false_done`	whole number	Times it reported work as done or tested when it was not
`invented_reasons`	whole number	Times it gave a reason, a limit or a fact that was not true
`unrequested_work`	whole number	Times it reported a problem that was not there or did work nobody asked for

A count that was not kept is left out, and the report shows "not counted".

The answers file

What `--answers FILE` reads and `--save-answers FILE` writes. Every part may be left out.

```
{
  "paths": {"decision_records": "docs/decisions"},
  "not_applicable": ["6.5"],
  "survey": {"7.1": "yes", "8.1": "no", "12.1": "declined"},
  "beliefs": {"9.1": "yes", "9.2": "no", "6.3": "unknown"},
  "interview": {"9.6": "shown", "7.3": "claimed", "6.4": "no", "8.3": "declined"},
  "model_conduct": [{"model": "model-a 1.0", "period": "2026-10", "rule_breaks": 4,
    "false_done": 1}]
}
```

- `survey`: the 18 survey checks; `yes`, `no` or `declined`.
- `beliefs`: the 35 scan checks, what the client believes is in place; `yes`, `no` or `unknown`.
- `interview`: the 44 interview checks, and any scan check the tool cannot read on the platform; `shown` (the assessor saw the artifact), `claimed` (described, not shown), `no` or `declined`.
- `paths.decision_records`: the folder of decision records, inside the repository.
- `model_conduct`: at most 12 rows, 1 for each model and period, with the fields of "Each of `model_conduct`" above. The questionnaire at the keyboard does not ask for them.
- `not_applicable`: only `6.5` and `14.6` are accepted: `6.5` for a client who names no crucial function, `14.6` for a client whose agents run no command on another machine.

The same file on every system

The results file has no field for the operating system. A location in `checks[].locations` and `read.paths` is the name the vendor documents, not a path on your disk: `~/.claude/settings.json`, `%APPDATA%\Code\User\settings.json`, `C:\Program Files\ClaudeCode\managed-settings.json`. A file inside the repository is shown relative to the repository with `/` on every system. On Windows the answers file may start with a UTF-8 byte-order mark, and `paths.decision_records` may be written with `\`.

Uploading the file

`--upload` and `--upload-file` send this file, as it is, in 1 `POST` with the content type `application/json`. Nothing else is sent. The file is at most 262,144 bytes.

The results site accepts the file only when all of these hold:

- `schema_version` is "1.0" and `framework.catalog_version` is the site's own catalog version;
- `checks` holds the catalog's 97 checks, each with a result the schema allows;
- the score and level of each dimension and of `overall`, recomputed by the site from `checks`, equal the ones in the file;
- `said_but_not_found` and `in_place_but_not_known` equal the lists the site recomputes;
- `next_actions` holds at most 10 entries, each naming a check that is not passed.

The site keeps a copy that holds only the fields this document names, without `machine`: a file that carries `machine` is accepted and stored without it. The site replaces `limits` with its own catalog's text and drops any field this document does not name.

The site answers with JSON: `claim_code` (32 characters), `claim_link`, `expires` (the UTC time the code stops working), `claim_days` (the number of days the code works) and `sha256` (the SHA-256 of the bytes the site received, in hexadecimal). The tool prints the SHA-256 of the file before it sends and warns when the site reports another value. The site keeps the value and shows it with the report, so a person who holds the file can see that the saved result is that file. The value does not show that the answers in the file are true. A refusal is JSON with `error`, a sentence that quotes nothing from the file.

The published benchmark figures

`benchmark.json`, schema "1.0": what the results site publishes at `/benchmark.json` and what `--benchmark` reads. It is computed at each request from the results whose owners share them, 1 per member and coding agent. It holds no result, no owner, no file name and no reason.

Field	Type	Meaning
`schema`	string	"1.0"
`computed`	string	UTC time of the computation
`minimum_group`	number	11: no figure is given for fewer results
`catalog_version`	string	The site's catalog version
`groups`	list of objects	The first is every shared result; then 1 for each coding agent, industry, size, kind of ownership and funding stage that has at least 1 shared result

Each of `groups`:

Field	Type	Meaning
`cut`	string	`all`, `agent`, `industry` or `size`
`value`	string	"" for `all`; a platform id; an ISIC Rev. 5 section letter; or a size band (`1-9`, `10-49`, `50-249`, `250-999`, `1000+`)

Field	Type	Meaning
`label`	string	The group's name
n	number	Shared results in the group
`withheld`	true	Present when the group has at least 11 results and its figures are not given, because the rest of the shared results number 1 to 10
`overall`	object	Present only when figures are given: n, and `q1`, `median`, `q3` of the overall score to 1 decimal place
`dimensions`	list of 16 objects	Present only with `overall`: `number`, n (results in which the dimension is rated), and `q1`, `median`, `q3` when at least 11 are rated and the rated rest of the shared results is 0 or at least 11

A quartile is the value at position (n - 1) x p of the sorted scores, counted from 0, with linear interpolation between the 2 nearest. `results.json` is unchanged by the benchmark: its schema stays "1.0".

9. Moving a practice from Written to Checked to Enforced

The 3 kinds: a Written rule is a sentence the agent reads; a Checked rule is a mechanism that refuses the result after the agent has acted; an Enforced rule is a mechanism that refuses the action before it happens. A practice can exist at all 3 at once. Take "run the tests before you finish":

- Written: a command line or heading in the instruction file that names tests (check 6.1).
- Checked: a git hook that is not a sample, or a workflow file (check 6.2).
- Enforced: a hook on the end of the agent's turn (check 6.3).

Checked, on every platform. 2 of the 3 level 2 scan checks read the repository, so they are the same for all 7 platforms: any file in `.git/hooks/` whose name does not end in `.sample`, or a file in `.github/workflows/` (6.2), and a `CODEOWNERS` file in 1 of the 3 locations in section 3 (7.2). The third, 16.4, reads whether a hook is registered on denied or failed tool calls: `PermissionDenied` on Claude Code, which runs when auto mode denies a call, and `postToolUseFailure` on Cursor. No such event is documented for the other 5 platforms, so there 16.4 is shown to an assessor. The other level 2 checks are shown to an assessor: the script or gate that refuses, with its refusal.

Enforced, by platform. The level 3 scan checks are 1.4, 4.5, 4.6, 5.2, 6.3, 9.1 to 9.5, 12.2, 13.3 to 13.6, 14.1, 14.3, 15.11 and 16.7. What each vendor documents, from the research files 05 section 6, 10 section 2 and 11 section 2, and what the scan therefore reads:

Platform	Mechanism the vendor documents	Checks it can pass in the scan
Claude Code	Hooks under the key `hooks` in `.claude/settings.json`, `.claude/settings.local.json` and `~/.claude/settings.json`, on events that include `UserPromptSubmit`, `PreToolUse` and `Stop`; a hook that exits with code 2 blocks the action. Permission rules in the same files	1.4 (`UserPromptSubmit`), 4.6 (`PreToolUse` on Agent), 5.2 (`PreToolUse` on Bash), 6.3 (`Stop`), 9.4 (`PreToolUse`), 9.5 (`onFailure` set to `block` on every `PreToolUse` hook), 14.1 (`PreToolUse` on Write or Edit), 16.7 (`SubagentStop` or `TaskCompleted`), 9.3 (the `permissions` lists)
Claude Code, cost settings	`autoCompactWindow`; `effortLevel`; `CLAUDE_CODE_SUBAGENT_MODEL`, or `model` in an agent definition; `promptCacheTtl` or `CLAUDE_CODE_PROMPT_CACHE_TTL`; `tools` or `disallowedTools` in an agent definition	13.3, 13.4, 13.5, 13.6, 14.3
Claude Code, keys taken from settings files in use (01 sections 4 and 13), not from the vendor page read	`sandbox.enabled`; `sandbox.network.allowedDomains`; a model named with a version number in an `env` entry or in an agent definition's `model` front matter; `sandbox.allowUnsandboxedCommands`	9.1, 9.2, 4.5, 15.11

Platform	Mechanism the vendor documents	Checks it can pass in the scan
Cursor	`.cursor/hooks.json` and `~/.cursor/hooks.json`, with events that include `preToolUse`, `beforeShellExecution`, `beforeMCPExecution`, `beforeReadFile` and `stop`; a hook fails open unless it has `failClosed: true`. Allow and deny lists in `~/.cursor/cli-config.json` and `.cursor/cli.json`; deny wins. An enterprise hooks file at `/Library/Application Support/Cursor/hooks.json` (macOS), `/etc/cursor/hooks.json` (Linux) or `C:\ProgramData\Cursor\hooks.json` (Windows). Agent files in `.cursor/agents/` with `model` and `readonly`	5.2 (`beforeShellExecution`), 6.3 (`stop`), 16.7 (`subagentStop`), 9.4, 9.5 (`failClosed: true` on every hook entry), 9.3, 12.2, 13.5, 14.3
Codex CLI	`sandbox_mode` (`read-only`, `workspace-write`, `danger-full-access`) and `approval_policy` (`on-request`, `never`, `granular`) in `~/.codex/config.toml` and in the project's `.codex/config.toml`, which is loaded only for a trusted project. Hooks in `.codex/hooks.json` and `~/.codex/hooks.json`, on `UserPromptSubmit`, `PreToolUse` and `Stop`. The keys `model_auto_compact_token_limit`, `model_reasoning_effort` and `agents.default_subagent_model`	9.1 (`sandbox_mode` is not `danger-full-access`), 9.3 (`approval_policy` is not `never`), 1.4, 6.3, 16.7 (`SubagentStop`), 9.4, 13.3, 13.4, 13.5. The inline `[hooks]` table of `config.toml` is not read
VS Code with Copilot	The setting keys `chat.agent.sandbox.enabled`, `chat.tools.terminal.autoApprove` and `chat.tools.global.autoApprove`, in the workspace's `.vscode/settings.json` and in your user `settings.json`. The keys `chat.agent.networkFilter`, `chat.agent.allowedNetworkDomains` and `chat.permissions.default` are documented without their values, so the scan does not grade them. Hooks in `.github/hooks/*.json` and `~/.copilot/hooks/`, on `userPromptSubmitted`, `preToolUse` and `agentStop`. Agent files in `.github/agents/` with `model` and `tools`. On Linux, a policy file at `/etc/vscode/policy.json`	1.4, 6.3, 16.7 (`subagentStop`; the page read does not say that this hook can refuse the end of the work), 9.4, 13.5, 14.3; 9.1 (`chat.agent.sandbox.enabled` is true); 9.3 (`chat.tools.terminal.autoApprove` is set and `chat.tools.global.autoApprove` is not true); 12.2 on Linux. Show the network settings to the assessor for 9.2
Xcode	Settings, Intelligence, Permissions (Allowed Commands, Allowed Tools). A device-management key, `CodingAssistantAllowExternalIntegrations`, turns the assistant off when set to `false`	None in the scan: nothing on disk is documented. Show the settings to the assessor
Antigravity	Rule files with 4 activation modes (`always_on`, `model_decision`, `glob`, `manual`). The page read states nothing on settings, permissions or hooks	None at level 3. The rule files pass the Written checks 4.1 and 4.3
Grok Build	A configuration file, `~/.grok/config.toml`; its keys are not stated on the pages read	None. The scan does not open the file

On Claude Code, check 12.2 (organization policy) passes when a managed settings file exists at the vendor's path for your system; when none exists it is shown to the assessor, because the vendor also delivers policy without a file. On Codex CLI, 12.2 reads `/etc/codex/requirements.toml` and `/etc/codex/managed_config.toml` on macOS and Linux, and `%ProgramData%\OpenAI\Codex\requirements.toml` and `~/.codex/managed_config.toml` on Windows. Check 9.5 fails on Claude Code unless every `PreToolUse` hook has `onFailure` set to `block`, because the vendor states that a hook's error or timeout otherwise lets the call go ahead. A check the scan cannot read still counts: shown to the assessor it is a Pass; described and not shown it is Claimed.

10. What the tool does not do

- It shows that a control is present, not that it works. A registered hook passes whether it guards 1 command or 100; the tool never runs a hook.
- Checks 4.2, 6.1 and 9.7 match headings, command lines and a few words. An instruction file that covers the same ground in other words fails them.
- It reads the Codex configuration file with its own small reader, which takes scalar keys at the top level and inside tables, and the names of tables, and nothing else. The keys the Codex checks read are of those kinds.
- It reads `.git/hooks/` only. A repository whose `.git` is a file (a linked worktree), or whose hooks live elsewhere through `core.hooksPath`, fails 6.2 unless it has a workflow file.
- Branch protection is not on disk. When no `CODEOWNERS` file is found, check 7.2 is shown to the assessor.
- It finds the Claude Code project folder (memory and session records) from the repository's path on macOS and Linux. No page read gives the folder's name on Windows, so there those 2 checks go to a person.
- No person has run it on a Linux or Windows workstation with real agent files. What it does there is tested by simulation and by the suite on GitHub's Linux and Windows machines.
- Check 13.12 reads usage telemetry in the user settings file only; a variable set in the shell or in managed settings is not seen. Checks 13.2, 13.6, 13.12, 14.1 and 15.11 are scanned on Claude Code only.
- Check 13.2 is graded on Claude Code only, with the 200-line figure Anthropic gives for its own instruction file. No other vendor's page read gives a figure, so elsewhere the check is Unknown whatever you answer.
- All 11 checks of dimension 15 come from the process of Kyberna alone.
- In dimension 16 the scan reads 2 things: whether a hook is registered on denied or failed tool calls (16.4), and whether one is registered on the end of a started agent's work or on a task being marked complete (16.7). Check 6.3 reads the hook on the end of the main turn, so 1 hook never passes both 6.3 and 16.7. The tool opens no session record and counts no failure of any model: the table of counts for each model under the dimensions table holds the numbers you put under `model_conduct` in the answers file, and they change no grade. No vendor page read documents where a log of refused calls or a written model policy is kept, so checks 16.5, 16.8 and 16.9 are shown to an assessor.
- It assesses an agent only when the repository holds a file of that agent. `.agents/` is a marker of Antigravity and also the folder where Codex CLI keeps skills, so a Codex CLI repository with skills gets an Antigravity result as well; `--platform` assesses 1.
- The upload sends 1 file to 1 address over HTTPS. The tool does not sign the file: a key shipped inside a program that runs on your machine can be read by the person who runs it, so a signature made with it would prove nothing. The results site recomputes the score from the check results and refuses a file whose score does not match, so a score cannot be edited alone; a person who edits the check results gets the score those results give. The tool prints the SHA-256 of the file before it sends, and the site shows the same value with the saved result: equal values show that the saved result is the file the tool wrote, and nothing about whether its answers are true.
- No study read for the framework shows that a higher level brings faster delivery or better code. The result states which controls exist and can be shown.

Sources

Files are in the `research` folder, except 06, which is `framework/06-framework.md`: 01 is `01-kyberna-process.md`, 03 is `03-agent-practice.md`, 05 is `05-backend-options.md`, 10 is `10-token-and-cost.md`, 11 is `11-harness-features.md`, 12 is `12-process-guards.md`, 15 is `15-model-conduct.md`. "Tool" is the `tool` folder.

Statement	Source
Section 1	Tool `README.md`; tool `tests/test_privacy.py`
Section 2: commands and output	Tool `runs/synthetic-sample/dry-run.txt` and `runs/synthetic-sample/console.txt`, included by `briefings/build.py`; `assessments/kyberna/paper-test.md`, "The tool's run"

Statement	Source
Section 2: options	Tool `README.md`; tool `assessment/cli.py`
Section 3: counts and locations	Generated from tool `assessment/data/platforms.json`; the same table is 06 section 4.1
Section 3: where the locations come from	05 section 6, with each vendor's pages below; GitHub's pages for `CODEOWNERS` and workflow files, named in 06 section 4.1
Section 3: what is read in those locations	06 section 7; tool `README.md`
Section 4	Tool `EXPECTATIONS.md`, lines E1, E3, E6, E9 to E14, E17 and E34 to E89; test run of 2026-11-14
Section 5	Generated from tool `assessment/data/catalog.json`; the same rows are 06 section 4.2, where each check's own source is given
Section 6	06 sections 5.1 and 5.2
Section 7	06 sections 5.3, 5.4 and 5.5; tool `runs/kyberna/results.json`
Section 8	Tool `SCHEMA.md`, included by `briefings/build.py`
Section 9: the 3 kinds	06 section 2
Section 9: Claude Code	05 section 6; 10 section 2; 11 section 2; 12 section 2; https://code.claude.com/docs/en/claude-directory, https://code.claude.com/docs/en/hooks, https://code.claude.com/docs/en/model-config.md, https://code.claude.com/docs/en/prompt-caching.md and https://code.claude.com/docs/en/sub-agents
Section 9: Cursor	05 section 6; https://cursor.com/docs/agent/hooks and https://cursor.com/docs/cli/reference/permissions; 11 section 2, https://cursor.com/docs/subagents
Section 9: Codex CLI	05 section 6; 11 section 2; https://learn.chatgpt.com/docs/config-file/config-reference and https://learn.chatgpt.com/docs/hooks.md
Section 9: VS Code with Copilot	05 section 6; https://code.visualstudio.com/docs/agents/run/approvals; 11 section 2, https://code.visualstudio.com/docs/copilot/customization/hooks and https://code.visualstudio.com/docs/copilot/customization/custom-agents
Section 9: Xcode	05 section 6, which cites 3 pages of Apple's: https://developer.apple.com/documentation/xcode/extending-and-customizing-agents.md, https://developer.apple.com/documentation/xcode/setting-up-coding-intelligence.md and https://developer.apple.com/documentation/xcode/writing-code-with-intelligence-in-xcode.md
Section 9: Antigravity	05 section 6; https://antigravity.google/docs/rules
Section 9: Grok Build	05 section 6; https://docs.x.ai/build/overview
Section 9: dimension 16	15 parts B and C; https://code.claude.com/docs/en/hooks, https://learn.chatgpt.com/docs/hooks, https://cursor.com/docs/agent/hooks and https://code.visualstudio.com/docs/copilot/customization/hooks
Section 10	06 section 8; 15 part C.4; tool `assessment/scan.py`; 10 section 2 (the 200-line figure, https://code.claude.com/docs/en/costs; telemetry, https://code.claude.com/docs/en/monitoring-usage.md)