Custom instructions with AGENTS.md

Codex reads `AGENTS.md` files before doing any work. By layering global guidance with project-specific overrides, you can start each task with consistent expectations, no matter which repository you open.

How Codex discovers guidance

Codex builds an instruction chain when it starts (once per run; in the TUI this usually means once per launched session). Discovery follows this precedence order:

1. **Global scope:** In your Codex home directory (defaults to `~/.codex`, unless you set `CODEX_HOME`), Codex reads `AGENTS.override.md` if it exists. Otherwise, Codex reads `AGENTS.md`. Codex uses only the first non-empty file at this level.
2. **Project scope:** Starting at the project root (typically the Git root), Codex walks down to your current working directory. If Codex cannot find a project root, it only checks the current directory. In each directory along the path, it checks for `AGENTS.override.md`, then `AGENTS.md`, then any fallback names in `project_doc_fallback_filenames`. Codex includes at most one file per directory.
3. **Merge order:** Codex concatenates files from the root down, joining them with blank lines. Files closer to your current directory override earlier guidance because they appear later in the combined prompt.

Codex skips empty files and stops adding files once the combined size reaches the limit defined by `project_doc_max_bytes` (32 KiB by default). For details on these knobs, see Project instructions discovery. Raise the limit or split instructions across nested directories when you hit the cap.

Create global guidance

Create persistent defaults in your Codex home directory so every repository inherits your working agreements.

1. Ensure the directory exists:

   ```
   mkdir -p ~/.codex
   ```

2. Create `~/.codex/AGENTS.md` with reusable preferences:

   ```
   # ~/.codex/AGENTS.md

   ## Working agreements

   - Always run `npm test` after modifying JavaScript files.
   - Prefer `pnpm` when installing dependencies.
   - Ask for confirmation before adding new production dependencies.
   ```

3. Run Codex anywhere to confirm it loads the file:

   ```
   codex --ask-for-approval never "Summarize the current
           instructions."
   ```

   Expected: Codex quotes the items from `~/.codex/AGENTS.md` before proposing work.

Use `~/.codex/AGENTS.override.md` when you need a temporary global override without deleting the base file. Remove the override to restore the shared guidance.

Layer project instructions

Repository-level files keep Codex aware of project norms while still inheriting your global defaults.

1. In your repository root, add an `AGENTS.md` that covers basic setup:

   ```
   # AGENTS.md

   ## Repository expectations

   - Run `npm run lint` before opening a pull request.
   - Document public utilities in `docs/` when you change behavior.
   ```

2. Add overrides in nested directories when specific teams need different rules. For example, inside `services/payments/` create `AGENTS.override.md`:

   ```
   # services/payments/AGENTS.override.md

   ## Payments service rules

   - Use `make test-payments` instead of `npm test`.
   - Never rotate API keys without notifying the security channel.
   ```

3. Start Codex from the payments directory:

```
codex --cd services/payments --ask-for-approval never "List the
        instruction sources you loaded."
```

Expected: Codex reports the global file first, the repository root `AGENTS.md` second, and the payments override last.

Codex stops searching once it reaches your current directory, so place overrides as close to specialized work as possible.

Here is a sample repository after you add a global file and a payments-specific override:

<FileTree class="mt-4" tree={[{ name: "AGENTS.md", comment: "Repository expectations", highlight: true, }, { name: "services/", open: true, children: [{ name: "payments/", open: true, children: [{ name: "AGENTS.md", comment: "Ignored because an override exists", }, { name: "AGENTS.override.md", comment: "Payments service rules", highlight: true, }, { name: "README.md" },], }, { name: "search/", children: [{ name: "AGENTS.md" }, { name: "…", placeholder: true }], },], },]} />

Customize fallback filenames

If your repository already uses a different filename (for example `TEAM_GUIDE.md`), add it to the fallback list so Codex treats it like an instructions file.

1. Edit your Codex configuration:

   ```
   # ~/.codex/config.toml
   project_doc_fallback_filenames = ["TEAM_GUIDE.md", ".agents.md"]
   project_doc_max_bytes = 65536
   ```

2. Restart Codex or run a new command so the updated configuration loads.

Now Codex checks each directory in this order: `AGENTS.override.md`, `AGENTS.md`, `TEAM_GUIDE.md`, `.agents.md`. Filenames not on this list are ignored for instruction discovery. The larger byte limit allows more combined guidance before truncation.

With the fallback list in place, Codex treats the alternate files as instructions:

<FileTree class="mt-4" tree={[{ name: "TEAM_GUIDE.md", comment: "Detected via fallback list", highlight: true, }, { name: ".agents.md", comment: "Fallback file in root", }, { name: "support/", open: true, children: [{ name: "AGENTS.override.md", comment: "Overrides fallback guidance", highlight: true, }, { name: "playbooks/", children: [{ name: "…", placeholder: true }], },], },]} />

Set the `CODEX_HOME` environment variable when you want a different profile, such as a project-specific automation user:

```
CODEX_HOME=$(pwd)/.codex codex exec "List active instruction sources"
```

Expected: The output lists files relative to the custom `.codex` directory.

Verify your setup

- Run `codex --ask-for-approval never "Summarize the current instructions."` from a repository root. Codex should echo guidance from global and project files in precedence order.
- Use `codex --cd subdir --ask-for-approval never "Show which instruction files are active."` to confirm nested overrides replace broader rules.
- Check `~/.codex/log/codex-tui.log` (or the most recent `session-*.jsonl` file if you enabled session logging) after a session if you need to audit which instruction files Codex loaded.
- If instructions look stale, restart Codex in the target directory. Codex rebuilds the instruction chain on every run (and at the start of each TUI session), so there is no cache to clear manually.

Troubleshoot discovery issues

- **Nothing loads:** Verify you are in the intended repository and that `codex status` reports the workspace root you expect. Ensure instruction files contain content; Codex ignores empty files.
- **Wrong guidance appears:** Look for an `AGENTS.override.md` higher in the directory tree or under your Codex home. Rename or remove the override to fall back to the regular file.
- **Codex ignores fallback names:** Confirm you listed the names in `project_doc_fallback_filenames` without typos, then restart Codex so the updated configuration takes effect.
- **Instructions truncated:** Raise `project_doc_max_bytes` or split large files across nested directories to keep critical guidance intact.
- **Profile confusion:** Run `echo $CODEX_HOME` before launching Codex. A non-default value points Codex at a different home directory than the one you edited.

Next steps

- Visit the official AGENTS.md website for more information.
- Review Prompting Codex for conversational patterns that pair well with persistent guidance.