DocCompiler.ai

USER GUIDE

Codex User Guide

Installation, Usage & Quick Reference

A Cyber Panda Solutions LLC product

Version 3.1 2026-05-13 @ f017a23 Generated by DocCompiler.ai

Always-updated version at doccompiler.ai/openai/codex

Source code analysis by Claude Opus 4.6 | 1M token context

About This Document

What Is This?

This is a comprehensive, source-code-verified **User Guide** for **Codex**, generated by DocCompiler.ai using Claude Opus 4.6 with 1M-token context. Every command, configuration option, and behavior documented here was verified against the actual source code.

The Definitive Reference

This document is designed to be the single source of truth for Codex. Rather than piecing together scattered README fragments, outdated wiki pages, forum posts, and version-specific screenshots, you can rely on this as a verified, structured reference backed by automated code review.

Optimized for AI Assistants

This document is structured for consumption by AI assistants. Feed this PDF into your preferred LLM for accurate, grounded answers about Codex — instead of having the model guess or hallucinate from incomplete context.

Version & Updates

Generated on **2026-05-13** (version **3.1**). This document reflects the source code at commit `f017a23`. The always-updated version is permanently available at **doccompiler.ai/openai/codex**. Do not print or download for offline use — this document is updated when the source code changes.

Our Mission

DocCompiler.ai aims to be the definitive wiki for the world's most important open-source projects. Every document is overkill by design: exhaustive, fact-checked, and built to replace the scattered tribal knowledge that slows teams down.

Table of Contents

OpenAI Codex -- User Guide

1. Quick Start

OpenAI Codex is an agentic coding assistant that runs in your terminal or as a desktop application. It reads your codebase, understands context, executes commands in a sandboxed environment, applies file changes, and helps you build, debug, and refactor software through natural-language conversation.

Key features:

- Interactive terminal UI (TUI) with streaming responses and inline code rendering
- Non-interactive batch mode (`codex exec`) for CI/CD and scripting
- Platform-native sandboxing (Linux Landlock/bubblewrap, macOS Seatbelt, Windows restricted tokens)
- Multi-agent orchestration with collaboration mode
- MCP (Model Context Protocol) server integration for extensible tool access
- Session persistence -- resume or fork previous conversations
- Structured output via JSON Schema for automation pipelines
- Plugin and skill marketplace
- Python and TypeScript SDKs for programmatic use

Prerequisites:

- Node.js 16+ (for npm install) **or** a supported OS for standalone install
- An OpenAI account (ChatGPT or API key)

Fastest install:

```bash
npm install -g @openai/codex
```

First run:

```bash
codex login
codex "Explain this codebase"
```

```text
Codex ➜ Analyzing repository structure...
Found 42 source files across 8 directories.
This is a Python web application using Flask with...
```

> **Tip**
>
> Run `codex` with no arguments to enter the interactive TUI -- the richest experience with streaming output, approval dialogs, and session history.

What to try next: See Installation for all methods, Configuration for customization, or jump to Core Usage for task-based workflows.

2. Installation

System Requirements

Requirement	Detail
OS	macOS (Intel/Apple Silicon), Linux (x64/arm64), Windows (x64/arm64)
Node.js	16+ (npm install method only)
Git	Recommended for session tracking and code review features
Network	Required for authentication and model API calls

Install Methods

■ npm (recommended)

```bash
npm install -g @openai/codex
```

■ Standalone (macOS/Linux)

```bash
curl -fsSL https://raw.githubusercontent.com/openai/codex/main/scripts/install/install.sh | bash
```

■ Standalone (Windows)

```powershell
irm https://raw.githubusercontent.com/openai/codex/main/scripts/install/install.ps1 | iex
```

■ Specific Version

```bash
# npm
npm install -g @openai/codex@0.6.0

# Standalone (macOS/Linux)
curl -fsSL https://raw.githubusercontent.com/openai/codex/main/scripts/install/install.sh | bash -s -- --rel

# Standalone (Windows)
irm https://raw.githubusercontent.com/openai/codex/main/scripts/install/install.ps1 | iex -Release 0.6.0
```

> **Important**
> The standalone installer verifies SHA-256 checksums on every download. If verification fails, the install aborts. Do not bypass this check.

Post-install verification:

```bash
codex --version
```

```text
codex 0.121.0
```

Platform Notes

macOS: The standalone installer places the binary at `~/.codex/packages/standalone/releases/<version>/` and creates a symlink at `~/.local/bin/codex`. It automatically adds `~/.local/bin` to your PATH via shell profile files (`.zshrc`, `.bash_profile`, etc.).

Linux: Same layout as macOS. The installer detects existing brew/npm-managed installs and offers to uninstall them to avoid conflicts.

Windows: Installs to `%LOCALAPPDATA%\Programs\OpenAI\Codex\bin\` using NTFS junctions for version management. Updates the User-scope `PATH` registry value.

WSL: Codex works under WSL2. WSL1 is **not supported** -- the sandbox requires user namespaces, which WSL1 cannot create.

> **Warning**
>
> On Windows in WSL mode, Codex may use the Windows `CODEX_HOME` path inside WSL, creating worktrees on `/mnt/c` instead of the WSL filesystem. This is a known issue (#13762). Set `CODEX_HOME` explicitly in your WSL environment to work around this.

Install Troubleshooting

Error: `command not found: codex` after install

Cause: The install directory is not in your `PATH`.

Solution:

1. Restart your terminal or run `source ~/.bashrc` (or `.zshrc`)
2. Verify: `echo $PATH | tr ':' '\n' | grep -i codex`
3. If missing, add manually: `export PATH="$HOME/.local/bin:$PATH"`

Error: npm permission errors on Linux/macOS

Cause: Global npm directory requires elevated permissions.

Solution:

```bash
mkdir -p ~/.npm-global
npm config set prefix '~/.npm-global'
export PATH="$HOME/.npm-global/bin:$PATH"
npm install -g @openai/codex
```

Error: PowerShell execution policy blocks install.ps1

Cause: Windows execution policy prevents running unsigned scripts.

Solution:

```powershell
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

Error: SHA-256 verification fails

Cause: Corrupt download or network interference.

Solution: Retry the install. If it persists, download the release manually from GitHub and compare checksums.

Error: Conflict with existing Codex install (npm/brew/bun)

Cause: Multiple package managers installed different versions.

Solution: The standalone installer detects these conflicts automatically and offers to uninstall the conflicting version. Follow the prompts.

3. Configuration

Codex uses TOML configuration stored at `~/.codex/config.toml`. Project-level overrides can be placed in `.codex/config.toml` within any repository.

Minimal Config Example

```toml
# ~/.codex/config.toml
model = "gpt-5"
approval_policy = "on-request"
sandbox_mode = "workspace-write"
```

Common Options

Key	Type	Default	Description
`model`	String	(server default)	Model to use (e.g., `"gpt-5"`, `"gpt-4o"`)
`approval_policy`	String	`"on-request"`	When to ask before executing: `never`, `on-request`, `on-failure`, `untrusted`
`sandbox_mode`	String	`"read-only"`	Sandbox level: `read-only`, `workspace-write`, `danger-full-access`
`profile`	String	--	Active named profile from `[profiles]`

`personality`	String	--	Model personality preset
`service_tier`	String	--	`"fast"` or `"flex"`
`web_search`	String	`"disabled"`	Web search: `disabled`, `cached`, `live`

Approval Policies

Policy	Behavior
`never`	Auto-approve everything (use with `workspace-write` sandbox)
`on-request`	Model decides when to ask; asks for destructive/uncertain ops
`on-failure`	Auto-approve first attempt; ask on failure
`untrusted`	Only auto-approve safe read-only operations

Sandbox Modes

Mode	Filesystem	Network
`read-only`	No writes anywhere	Restricted
`workspace-write`	Writes within project directory only	Configurable
`danger-full-access`	Unrestricted	Unrestricted

> **Warning**
>
> `danger-full-access` disables all sandbox protections. Only use this for trusted prompts in controlled environments.

Environment Variables

Variable	Purpose
`CODEX_HOME`	Override config/data directory (default: `~/.codex`)
`OPENAI_API_KEY`	API key (alternative to `codex login`)
`EDITOR` / `VISUAL`	External editor for TUI composition
`CODEX_TUI_RECORD_SESSION`	Enable session recording
`CODEX_TUI_SESSION_LOG_PATH`	Custom session log output path

> **Tip**
> Use CLI overrides (`-c key=value`) to change settings per-session without editing config files. For example: `codex -c model=gpt-4o -c sandbox_mode=workspace-write "Fix the tests"`.

For complete configuration reference, see the Admin Guide.

4. Core Usage

Interactive Mode (TUI)

Launch the interactive terminal UI:

bash

```
codex
```

The TUI provides:

- Real-time streaming of agent responses with markdown rendering
- Approval dialogs for command execution and file changes
- Session history with resume capability
- Multi-agent status indicators
- Rate limit visibility
- External editor integration (`$EDITOR`)

Common TUI flags:

Flag	Description
`--model`, `-m`	Select model
`--sandbox`, `-s`	Sandbox mode
`--full-auto`	Alias for `--sandbox workspace-write` with auto-approval
`--profile`, `-p`	Use named config profile
`--ask-for-approval`, `-a`	Override approval policy
`--add-dir DIR`	Add writable directory to sandbox
`--search`	Start in file search mode
`--no-alt-screen`	Disable alternate screen buffer
`--oss`	Use open-source model provider
`--local-provider NAME`	Use `lmstudio` or `ollama`

Start with a prompt:

bash

```
codex "Add error handling to the database module"
```

Attach images:

bash

```
codex --image screenshot.png "What's wrong with this UI?"
```

Non-Interactive Mode (Exec)

Run Codex as a batch command -- ideal for CI/CD, scripts, and automation:

bash

```
codex exec "Refactor the auth module to use async/await"
```

text

```
Codex ➜ Reading auth module...
Codex ➜ Applying changes to src/auth.js...
✓ Refactored 3 functions to async/await pattern
```

Key exec flags:

Flag	Description
`--full-auto`	No approval prompts, workspace-write sandbox
`--dangerously-bypass-approvals-and-sandbox` / `--yolo`	Skip all protections (externally-sandboxed environments only)
`--json`	Output all events as JSONL to stdout
`-o`, `--output-last-message FILE`	Write final agent message to file
`--output-schema FILE`	Enforce JSON Schema on response
`--ephemeral`	No session persistence
`--color MODE`	`always`, `never`, `auto`
`--skip-git-repo-check`	Allow running outside a git repo
`-C`, `--cd DIR`	Set working directory
`-m`, `--model MODEL`	Select model
`-s`, `--sandbox POLICY`	Sandbox mode
`-p`, `--profile NAME`	Config profile

Pipe input from stdin:

bash

```
cat error.log | codex exec "Diagnose this error"
```

> **Note**
> When stdin is piped, Codex appends the piped content as a `<stdin>` block alongside any positional prompt argument.

Structured output for automation:

bash

```
codex exec --output-schema schema.json "Analyze this codebase"
```

The agent's final response will conform to the JSON Schema in `schema.json`.

Code Review

Review changes against a branch, commit, or working tree:

bash

```
# Review uncommitted changes
codex review --uncommitted

# Review against a base branch
codex review --base main

# Review a specific commit
codex review --commit abc1234

# Custom review instructions
codex review --base main "Focus on security issues"
```

> **Tip**
> The `review` subcommand supports `--title` to provide commit context: `codex review --commit abc1234 --title "Add caching layer"`.

Authentication

Browser-based login (default):

bash

```
codex login
```

Opens your browser for ChatGPT OAuth authentication.

Device code flow (headless/SSH):

bash

```
codex login --device-auth
```

API key:

bash

```
printenv OPENAI_API_KEY | codex login --with-api-key
```

> **Important**
>
> Never pass API keys as command-line arguments. Always pipe from environment variables or stdin.

Log out:

bash

```
codex logout
```

5. Common Tasks

Resume a Previous Session

bash

```
# Show session picker
codex resume

# Resume the most recent session
codex resume --last

# Resume by session ID or name
codex resume "my-refactor-session"

# Resume and send a follow-up prompt
codex resume --last "Now add tests for those changes"

# Include non-interactive sessions in picker
codex resume --all --include-non-interactive
```

Fork a Session

Create a new branch of conversation from an existing session:

bash

```
codex fork --last
```

Manage MCP Servers

Connect external tools via the Model Context Protocol:

bash

```
# Add an MCP server (stdio transport)
codex mcp add my-server -- node path/to/server.js

# Add an MCP server (HTTP transport)
codex mcp add my-server --url https://mcp.example.com

# List configured servers
codex mcp list

# Remove a server
codex mcp remove my-server
```

```
# OAuth login for an HTTP server
codex mcp login my-server

# OAuth logout
codex mcp logout my-server
```

> **Tip**
>
> MCP servers configured in `config.toml` under `[mcp_servers]` are automatically available in every session. Use `codex mcp add` for quick per-session additions.

Run Codex as an MCP Server

Expose Codex itself as an MCP server for other tools:

bash

```
codex mcp-server
```

This starts a stdio-transport MCP server that other MCP clients can connect to.

Run Commands in a Sandbox

Test sandbox behavior without running the full agent:

bash

```
# macOS Seatbelt sandbox
codex sandbox macos -- ls /etc

# Linux Landlock sandbox
codex sandbox linux -- cat /etc/passwd

# Full-auto mode (writes allowed in cwd)
codex sandbox macos --full-auto -- python script.py

# Log sandbox denials on macOS
codex sandbox macos --log-denials -- make build
```

Use the Desktop App (macOS)

bash

```
# Launch Codex Desktop (downloads if needed)
codex app

# Open a specific workspace
codex app /path/to/project
```

Manage Feature Flags

bash

```
# List all feature flags
codex features list

# Enable a feature
```

```
codex features enable memory

# Disable a feature
codex features disable memory
```

> **Note**
>
> Feature flags can also be toggled with CLI globals: `codex --enable memory` or `codex --disable memory`.

Manage Plugins

bash

```
# Install a plugin from the marketplace
codex plugin install my-plugin

# List installed plugins
codex plugin list

# Remove a plugin
codex plugin uninstall my-plugin
```

Apply Agent Changes via Git

After a Codex agent session produces a diff:

bash

```
codex apply
```

This applies the most recent diff from the agent as a `git apply` operation.

Cloud Tasks (Experimental)

Browse and apply changes from Codex Cloud tasks:

bash

```
# List recent tasks
codex cloud list

# Show task details
codex cloud show TASK_ID

# Apply changes from a cloud task
codex cloud apply TASK_ID
```

Generate Shell Completions

bash

```
# Bash
codex completion bash &gt;&gt; ~/.bashrc

# Zsh
codex completion zsh &gt;&gt; ~/.zshrc

# Fish
```

```
codex completion fish &gt; ~/.config/fish/completions/codex.fish
```

Use the TypeScript SDK

typescript

```
import { Codex } from "@openai/codex-sdk";

const codex = new Codex();
const thread = codex.startThread({
  model: "gpt-5",
  sandboxMode: "workspace-write",
  workingDirectory: "/path/to/project",
});

// Single-turn
const turn = await thread.run("Add input validation to the API");
console.log(turn.finalResponse);

// Streaming
const { events } = await thread.runStreamed("Refactor the auth module");
for await (const event of events) {
  if (event.type === "item.completed") {
    console.log(event.item);
  }
}
```

Use the Responses API Proxy

Bridge local tools to the OpenAI Responses API:

bash

```
npm install -g @openai/codex-responses-api-proxy
codex-responses-api-proxy --help
```

6. Troubleshooting

Diagnostic Decision Tree

text

```
Issue?
|- Can't install -&gt; See Install Troubleshooting (Section 2)
|- Can't authenticate -&gt; Check login method below
|- Agent won't execute commands -&gt; Check approval_policy and sandbox_mode
|- Sandbox errors -&gt; Check platform sandbox support
|- Slow / hanging -&gt; Check rate limits and network
+- Wrong model -&gt; Check model config and account tier
```

Error: `I'm sorry, but I cannot assist with that request`

Cause: Content policy triggered by the prompt or workspace content. This was a known regression (#17615) where the model would refuse valid coding prompts.

Solution:

1. Rephrase the prompt to be more specific about the coding task
2. Ensure your workspace does not contain content that triggers safety filters
3. Update to the latest version -- this regression has been addressed in recent releases

Prevention: Keep prompts focused on specific coding tasks.

Error: Rate limit reached / usage throttled

Cause: Account usage limits for the current billing period, or session-level rate limiting (#18368).

Solution:

1. Wait for the rate limit window to reset (visible in the TUI status bar)
2. Use `service_tier = "flex"` in config for lower-priority but more available capacity
3. Reduce prompt complexity or break work into smaller sessions

Prevention: Monitor the rate limit indicators in the TUI.

Error: Token usage significantly increased after update

Cause: Model or prompt changes between versions may increase token consumption (#18345).

Solution:

1. Check `model_auto_compact_token_limit` in config -- lowering it triggers earlier context compaction
2. Use `--ephemeral` for one-off tasks that don't need session history
3. Pin a specific model version if consistency matters

Prevention: Review release notes before updating.

Error: Sandbox setup failed on WSL1

Cause: WSL1 cannot create user namespaces required by the Linux sandbox (bubblewrap).

Solution: Upgrade to WSL2:

```
powershell
```

```
 wsl --set-version &lt;distro&gt; 2
```

Prevention: Always use WSL2 for Codex on Windows.

Error: MCP server connection failed

Cause: MCP server process not found, wrong transport, or OAuth credentials expired.

Solution:

1. Verify the server command works independently
2. Check `codex mcp list` for server status
3. For OAuth servers, re-authenticate: `codex mcp login <server-name>`
4. Ensure the server implements the MCP protocol correctly

Prevention: Test MCP server commands before adding to config.

> **Caution**
>
> If `codex mcp add` fails with "invalid server name", ensure the name uses only letters, numbers, hyphens, and underscores. No spaces or special characters.

Error: `Computer Use plugin unavailable` on macOS

Cause: Known issue (#18258) where bundled plugin files exist but are not detected.

Solution: Update to the latest version. This issue is tracked and being addressed.

Error: Desktop app shows `Application Error`

Cause: Known regression (#18311) with mini window popout rendering on macOS.

Solution: Update to the latest version or restart the desktop app.

Error: Conversations disappear after update

Cause: Known issue (#18364) where bogus root-level `status` sessions flood recent local history, hiding older conversations.

Solution: Use `codex resume --all` to see all sessions regardless of recency filtering.

> **Warning**
>
> If Codex Desktop repeatedly starts full MCP stacks for new sessions/subagents, causing slowdown and memory pressure (#18333), restart the app and limit the number of concurrent MCP servers in your configuration.

Error: Automations not running on schedule

Cause: Known issue (#17840) with the automation scheduler.

Solution: Manually trigger automations via the UI while this issue is being resolved.

Error: `Recursive Context Poisoning` / false rate limits

Cause: Known issue (#17880) where Cloudflare/WAF interference triggers persistent history loss.

Solution:

1. Clear browser cookies and cache
2. Try a different network connection
3. If persists, contact OpenAI support

Prevention: Avoid aggressive VPN or proxy configurations that may trigger WAF rules.

Error: Windows app renders with black bars

Cause: Known issue (#18356) with light theme rendering on Windows.

Solution: Force Windows dark theme in system settings, or update to the latest version where this is fixed.

Error: Invalid tools schema with OpenRouter

Cause: Known issue (#18330) where Codex CLI sends tool schemas that OpenRouter rejects, while the IDE extension works with identical config.

Solution: Use the IDE extension for OpenRouter, or check for updates addressing this CLI-specific schema formatting issue.

7. FAQ

Q: How do I switch models mid-session?

You cannot switch models mid-conversation -- start a new session with the desired model. Use `codex -m gpt-4o` or set `model = "gpt-4o"` in config.

Q: Can I use Codex with local/open-source models?

Yes. Use `--oss` to enable OSS mode, or `--local-provider ollama` / `--local-provider lmstudio` to connect to a local model server. Configure custom providers in `config.toml` under `[model_providers]`. Set `CODEX_OSS_PORT` or `CODEX_OSS_BASE_URL` to customize the connection endpoint.

Q: How do I use Codex in CI/CD pipelines?

Use `codex exec` with `--full-auto --json --ephemeral`:

bash

```
codex exec --full-auto --json --ephemeral "Run lint and fix any issues" &gt; results.jsonl
```

Q: Where are my session histories stored?

In `~/.codex/` (or the directory specified by `CODEX_HOME`). Sessions are stored in a SQLite database with WAL mode enabled.

Q: How do I clear my conversation history?

bash

```
codex debug clear-memories
```

Q: Can I attach images to prompts?

Yes, use the `--image` / `-i` flag:

bash

```
codex --image screenshot.png "What's in this image?"
codex exec -i diagram.png,error.png "Debug these errors"
```

Multiple images can be comma-separated or specified with repeated `-i` flags.

Q: What's the difference between `--full-auto` and `--yolo`?

`--full-auto` enables `workspace-write` sandbox with auto-approvals -- the agent can write files in your project but is still sandboxed. `--dangerously-bypass-approvals-and-sandbox` (alias `--yolo`) removes **all**

protections -- only use it when Codex is already running inside an external sandbox (e.g., a Docker container). These two flags are mutually exclusive.

Q: How do I use Codex with a remote app server?

bash

```
codex --remote ws://192.168.1.100:4222 --remote-auth-token-env MY_TOKEN_VAR
```

The `--remote-auth-token-env` flag specifies the name of an environment variable containing the bearer token. Remote mode is only available for the interactive TUI -- other subcommands reject it.

Q: Does Codex support multiple working directories?

Yes. Use `--add-dir` to add additional writable directories alongside the workspace:

bash

```
codex --add-dir ../shared-lib "Update the shared types"
```

Q: How do I configure the Guardian auto-reviewer?

Set `approvals_reviewer = "guardian_subagent"` in `config.toml`. The Guardian is a risk-assessing sub-agent that automatically approves low-risk operations and escalates high-risk ones to you. It uses the `workspace-write` sandbox and `on-request` approval policy.

Q: Can I use Codex as an MCP server for other AI tools?

Yes. Run `codex mcp-server` to start Codex as a stdio-transport MCP server that any MCP client can connect to.

Q: How do I configure named profiles?

Add profiles to `config.toml` and switch with `--profile`:

toml

```
[profiles.fast]
model = "gpt-4o"
service_tier = "fast"

[profiles.thorough]
model = "gpt-5"
model_reasoning_effort = "high"
```

bash

```
codex --profile fast "Quick fix for this typo"
```

Q: What happens when context gets too long?

Codex automatically compacts conversation history when it approaches the context window limit. You can tune this with `model_auto_compact_token_limit` in config. Manual compaction is also available in the TUI.

Q: How do I set per-project configuration?

Create a `.codex/config.toml` file in your project root. Project-level settings override user-level settings but are overridden by CLI flags.

> ***DocCompiler.ai** v3.1 | Upgraded source coverage | Generated 2026-05-13 | Source: `openai/codex` @ `f017a23`*
> *A Cyber Panda Solutions LLC product. This document is intentionally comprehensive -- built to be the complete, source-verified reference an AI or human needs without consulting other sources. **Do not print:** the*

always-current version lives at doccompiler.ai/openai/codex.