# Agenten-Workflows und Rollen im OpenAI Codex CLI

## Rollen und Agenten-Workflows (Core/Advanced/Specialized)

Die vorliegende Dokumentation beschreibt ein **Rollen-Konzept mit „Core CLI Agents“, „Advanced CLI Agents“ und „Specialized CLI Agents“**, einschließlich spezifischer Unter-Rollen wie *Executor*, *Parser*, *Validator* usw. Wichtig ist: **Diese Rollentrennung ist** nicht **als getrennte, parallele „Sub-Agenten“ im Codex CLI implementiert**, sondern dient als **inhaltliche Anleitung in der AGENTS.md**. Offiziell unterstützt Codex CLI das Laden einer solchen AGENTS.md-Datei als zusätzlichen Kontext („Memory & project docs“), um dem KI-Modell Projektwissen und Leitplanken mitzugeben[[1]](https://github.com/openai/codex#:~:text=Memory%20%26%20project%20docs). Das CLI **sucht automatisch nach AGENTS.md** im Benutzerverzeichnis und Projektordner (global ~/.codex/AGENTS.md, dann im Repository-Root, dann im aktuellen Verzeichnis, **Merge-Reihenfolge von global zu lokal**[[1]](https://github.com/openai/codex#:~:text=Memory%20%26%20project%20docs)). Inhalte darin (z. B. Build-Anweisungen, Coding-Guidelines, Qualitätsregeln) beeinflussen das Verhalten des einzelnen Agenten.

**Nicht unterstützt** wird hingegen ein automatisches Orchestrieren mehrerer KI-Instanzen in parallelen Rollen (wie bei Anthropic Claude’s „parallel subagents“). **Codex CLI agiert als einzelner Agent**, der anhand der Prompt-Historie entscheidet, welche Schritte durchzuführen sind. Die in der Dokumentation definierten Rollen (*Core/Advanced/Specialized Agents*) sind also **kein offizielles Feature**, sondern eine inoffizielle Methode, um das **Verhalten über AGENTS.md zu strukturieren**. Codex CLI selbst führt *alle* Aufgaben seriell im Rahmen einer Sitzung aus – es gibt keine eingebaute Parallelisierung von Teilaufgaben. Ein Vergleich: **Claude Code nutzt tatsächlich parallele Subagenten**, die mit fokussierten Teilkontexten arbeiten und so Geschwindigkeit und Kontext-Effizienz gewinnen[[2]](https://www.reddit.com/r/ChatGPTCoding/comments/1m5uloy/are_there_any_real_benefits_in_using_terminalcli/#:~:text=Actually%2C%20Claude%20Code%20is%20much,secrete%20sauce%20of%20Claude%20Code)[[3]](https://www.reddit.com/r/ChatGPTCoding/comments/1m5uloy/are_there_any_real_benefits_in_using_terminalcli/#:~:text=Claude%20Code%20extensively%20uses%20subagents,search%20of%20Cline%2F%20RooCode%2F%20ZentaraCode). Codex CLI hat diese Funktion nicht; es handelt sich im Kern um einen monolithischen Agenten (ähnlich wie z. B. Cursor oder RooCLI ohne parallele Threads).

**Fazit zu den Rollen-Angaben:** Die Dokumentation ist insofern **korrekt**, als dass man in AGENTS.md Richtlinien und **verschiedene Aufgabentypen** definieren kann, die der Agent befolgen soll. Es ist jedoch **etwas irreführend**, diese als fest verdrahtete „Agenten“ zu bezeichnen. Offizielle Quellen erwähnen keine vordefinierten Rollenlisten – AGENTS.md ist frei gestaltbar. Beispielsweise empfiehlt eine Community-Referenz AGENTS.md-Datei, Informationen über Projektstruktur, Konventionen, Build-/Test-Befehle etc. bereitzustellen, damit der Agent konsistent im Projektkontext agiert[[4]](https://agentsmd.net/#:~:text=Agents,project%27s%20structure%2C%20conventions%2C%20and%20requirements)[[5]](https://agentsmd.net/#:~:text=2). Die im Dokument aufgeführten Core/Advanced/Specialized-Rollen sind also **Benutzer-Definitionen**, die Codex CLI nicht von Haus aus „kennt“, sondern die vom Entwickler manuell in die AGENTS.md eingefügt wurden. Offiziell wird lediglich unterstützt, **dass** solche Anleitungen eingelesen und **zum Prompt gemergt** werden[[1]](https://github.com/openai/codex#:~:text=Memory%20%26%20project%20docs). Es gibt keine Hinweis im offiziellen README oder Help Center, dass Codex von sich aus unterschiedliche Persönlichkeiten/Agentenmodi (außer den **Approval-Modi**, siehe unten) besitzt.

Wenn man also die Dokumentation validiert: **Richtig** ist, dass AGENTS.md die Möglichkeit bietet, **Rollen und Leitplanken zu definieren** und Codex dadurch in gewünschte Bahnen zu lenken. **Nicht offiziell** ist jedoch die **Terminologie „Core/Advanced/Specialized Agents“** – sie findet sich so nicht in OpenAIs eigener Dokumentation, sondern spiegelt eine inoffizielle Strukturierung wider. Entwickler können diese Strategie nutzen, müssen aber verstehen, dass Codex CLI intern weiterhin ein einzelner Agent bleibt, der diese Regeln verinnerlichen soll.

**Approval-Workflows:** Unabhängig von AGENTS.md unterscheidet Codex CLI *drei Betriebsmodi* bezüglich Ausführungsfreigaben, was in der Doku korrekt wiedergegeben ist. Die **“Suggest”-, “Auto-Edit”- und “Full-Auto”-Modi** werden offiziell unterstützt[[6]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Approval%20Modes)[[7]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Full%20Auto). – *Suggest* (Vorschlag, Standard): KI darf Code lesen und Änderungen/Kommandos vorschlagen, aber führt nichts ohne Zustimmung aus[[8]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Suggest%20). – *Auto-Edit*: KI darf Dateien selbst schreiben, fragt aber vor Shell-Befehlen um Erlaubnis[[9]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Auto%20Edit). – *Full-Auto*: KI darf schreiben **und** Kommandos ausführen, alles autonom in einer Sandbox ohne Netzwerk[[10]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Full%20Auto). Diese Modi sind durch Flags (--suggest, --auto-edit, --full-auto) oder während der Session per Befehl (/mode) umschaltbar[[11]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=How%20do%20I%20change%20approval,modes). Die Dokumentation beschreibt dies bereits richtig (inkl. Warnung bei nicht versioniertem Verzeichnis, etc.). Hier ist nichts Veraltetes festzustellen – diese Mechanik ist integraler Bestandteil und bleibt aktuell[[12]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Read%2C%20write%2C%20and%20execute%20commands,scoped%20to%20the%20current%20directory).

**Sandbox und Richtlinien:** Die Doku nennt zudem Sicherheitsleitplanken (kein Netz, keine geheimen Keys, etc.) und erwähnt, dass Codex keine Git-Pushs oder externen Aktionen ohne Anweisung vornehmen soll. Das entspricht dem beabsichtigten Gebrauch – **standardmäßig laufen Kommandos in einer** eingeschränkten Sandbox\*\* (Verzeichnis-skopiert, kein Netzwerk)[[13]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Read%2C%20write%2C%20and%20execute%20commands,scoped%20to%20the%20current%20directory). Diese Defaults können verfeinert werden: z.B. --sandbox workspace-write vs. read-only, --ask-for-approval on-request vs. never etc., was in der Doku mit Beispielen erläutert wurde. Das stimmt mit dem README überein (Details in „Choosing Codex’s level of autonomy“) und wird weiterhin offiziell unterstützt[[14]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=codex%20)[[15]](https://github.com/openai/codex#:~:text=The%20mechanism%20Codex%20uses%20to,policy%20depends%20on%20your%20OS).

Ein Detail aus der Doku betrifft *“trusted projects”*: Es wird angemerkt, man könne in ~/.codex/config.toml pro Projekt einen trust\_level setzen, um je nach Verzeichnis das Standardverhalten anzupassen. Das ist ein fortgeschrittenes Feature, das nicht prominent in der Haupt-Doku steht, aber **tatsächlich existiert** (in GitHub-Issues wird projects = { "/pfad/zum/repo" = { trust\_level = "trusted" } } erwähnt) – damit lässt sich z.B. definieren, dass in bestimmten vertrauten Projekten automatisch mehr Befehle ausgeführt werden dürfen, ohne jedes Mal bestätigen zu müssen[[16]](https://github.com/openai/codex/issues/2182#:~:text=,%7D%20%7D%20%5Bmodel_providers). Dies ist jedoch eher ein Randthema; die Kerndokumentation im Help Center erwähnt es nicht explizit.

**Zwischenfazit:** Die Dokumentation des Fragestellers ist hinsichtlich der Agenten-Workflows in dem Sinne **zutreffend**, dass sie einen sinnvollen Workflow mit Rollen als *Richtlinie* vorgibt. Es sollte aber klargestellt werden, dass diese Rollenaufteilung **nicht nativ vom Codex CLI gesteuert** wird. Eine mögliche Korrektur wäre, in der Doku deutlich zu machen: *„Codex CLI unterstützt AGENTS.md-Guidance. Die oben definierten Rollen (Core/Advanced/Specialized) dienen als inhaltlicher Leitfaden für den Agenten, sind aber keine getrennten KI-Instanzen.“* Zudem ist es hilfreich zu erwähnen, dass OpenAI’s Codex CLI (Stand heute) **keine parallelen Sub-Agenten à la Claude Flow** besitzt – alle vorgeschlagenen Schritte werden sequentiell vom einen Modell durchgeführt.

## Unterstützte Modelle im Codex CLI (OpenAI-Modelle, Azure, eigene Deployments)

Die Dokumentation nennt als Beispiele **„o3“, „gpt-4.1-mini“ und „gpt-5“** und fragt, ob und wie andere Modelle wie GPT-3.5, GPT-4o, Azure-Modelle oder eigene Deployments nutzbar sind. Hierzu der aktuelle Stand:

**Allgemein gilt:** Codex CLI kann *jedes OpenAI-Modell ansprechen*, das über die **“Responses API”** (Chat-Completions API) verfügbar ist[[17]](https://github.com/openai/codex#:~:text=Which%20models%20are%20supported%3F)[[18]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Which%20models%20does%20Codex%20use%3F). In der offiziellen FAQ heißt es: *„Any model available with Responses API“* wird unterstützt; man kann per CLI-Flag -m (--model) den gewünschten Modellnamen angeben[[17]](https://github.com/openai/codex#:~:text=Which%20models%20are%20supported%3F). Die Dokumentation des Fragestellers ist also insoweit korrekt, dass **nicht nur** die genannten Default-Modelle verwendbar sind, sondern z.B. explizit codex -m o3 für GPT-3.5-ähnliche Modelle oder -m gpt-4.1 für GPT-4.1 genutzt werden können.

**Was bedeuten „o3“ und „o4-mini“?** – Diese Bezeichnungen kamen mit dem Codex-Launch auf: *o3* ist ein intern codierter Name für ein fortgeschrittenes Modell der **GPT-3.5-Familie** (Optimierung auf „Reasoning“). *o4-mini* bezieht sich auf eine **kleinere GPT-4-Variante**, die schnell und kostengünstiger ist[[19]](https://visualstudiomagazine.com/articles/2025/04/17/mapping-openais-full-ai-suite-after-codex-cli-debut.aspx#:~:text=%2A%20New%20,to%20explore%2C%20use%2C%20and%20extend)[[20]](https://visualstudiomagazine.com/articles/2025/04/17/mapping-openais-full-ai-suite-after-codex-cli-debut.aspx#:~:text=execution%20Secure%20coding%2C%20automation%2C%20terminal,4o%20Code%20editing%2C%20AI). Tatsächlich hat OpenAI am 16. April 2025 neue Modelle eingeführt (oft *GPT-*o3 *und* GPT-*o4 Mini genannt) und kurz darauf die* *GPT-4.1-Familie* *veröffentlicht*[*[21]*](https://visualstudiomagazine.com/articles/2025/04/17/mapping-openais-full-ai-suite-after-codex-cli-debut.aspx#:~:text=On%20the%20same%20day%20that,Mini%2C%20adding%20to%20the%20mix)[*[22]*](https://visualstudiomagazine.com/articles/2025/04/17/mapping-openais-full-ai-suite-after-codex-cli-debut.aspx#:~:text=workflows%20in%20constrained%20environments%20GPT,cost%20queries)*. Zum Zeitpunkt der Codex-Veröffentlichung war* o4-mini *das Default-Modell des CLI*[*[17]*](https://github.com/openai/codex#:~:text=Which%20models%20are%20supported%3F)*, inzwischen jedoch bekommen Nutzer mit ChatGPT-Plus/Pro-Zugang standardmäßig das weiterentwickelte Modell* „GPT-5“ *zugewiesen*[*[23]*](https://github.com/openai/codex#:~:text=Run%20,%28Enterprise%20is%20coming%20soon)[*[18]*](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Which%20models%20does%20Codex%20use%3F)*. Hier ist eine kleine Begriffs-Verwirrung: OpenAI bezeichnet intern das neueste verfügbare reasoning-Modell für Plus-Nutzer als “GPT-5” (obwohl öffentlich von einem GPT-5-Modell i. e. S. noch nichts bekannt ist). Im Help Center steht ausdrücklich:* „By default Codex targets GPT-5 for fast reasoning“[*[18]*](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Which%20models%20does%20Codex%20use%3F)*. Das bedeutet:* *Wenn man sich via ChatGPT-Login beim CLI authentifiziert, nutzt Codex CLI das aktuell beste Modell (derzeit GPT-4.1 oder darüber, intern „gpt-5“ genannt) ohne zusätzliche API-Kosten*[*[23]*](https://github.com/openai/codex#:~:text=Run%20,%28Enterprise%20is%20coming%20soon)*.* *Verwendet man hingegen einen eigenen API-Schlüssel, könnte der Default je nach Version noch* o4-mini *sein – in jedem Fall kann man aber* *manuell das Modell setzen*\*.

**Nutzung von GPT-3.5 und GPT-4:** Selbstverständlich lässt sich GPT-3.5 Turbo einsetzen. Offizielle Beispiele nutzen dafür die Kurzbezeichnung o3 (steht gewissermaßen für die OpenAI-„3.x“-Reihe)[[18]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Which%20models%20does%20Codex%20use%3F). Alternativ kann man vermutlich auch gpt-3.5-turbo als Modellname angeben (das CLI leitet es an die API weiter). Ebenso ist GPT-4 verwendbar – konkret die GPT-4.1-Varianten, die 2025 neu eingeführt wurden. In der Dokumentation steht „gpt-4.1-mini“ – eine solche Schreibweise ist nicht offiziell, aber sie zielt wahrscheinlich auf **GPT-4.1 Mini** ab, was einer abgespeckten GPT-4.1-Version entspricht (schneller und günstiger, ~40% Kosten von vollem GPT-4.1)[[24]](https://visualstudiomagazine.com/articles/2025/04/17/mapping-openais-full-ai-suite-after-codex-cli-debut.aspx#:~:text=April%2014%2C%202025%201M,char%20limit%20Audiobooks). OpenAI nennt diese Modelle *GPT-4.1 Mini/Nano* etc., mit teils enormem Kontext (1 Mio Token) für große Aufgaben[[22]](https://visualstudiomagazine.com/articles/2025/04/17/mapping-openais-full-ai-suite-after-codex-cli-debut.aspx#:~:text=workflows%20in%20constrained%20environments%20GPT,cost%20queries). Das Codex CLI kann solche Modelle nutzen, solange sie über die API verfügbar sind. Beispielsweise könnte man --model gpt-4.1 wählen, um die *Flaggschiff*-Variante GPT-4.1 zu verwenden[[17]](https://github.com/openai/codex#:~:text=Which%20models%20are%20supported%3F). **GPT-4o**: In der Frage wird „gpt-4o“ genannt – vermutlich ist damit das **„GPT-4 (old/original)“** gemeint. In einigen Quellen wird GPT-4 (die ursprüngliche Version von 2023 mit 128k Kontext) als *GPT-4.0 bzw. GPT-4o* bezeichnet[[25]](https://visualstudiomagazine.com/articles/2025/04/17/mapping-openais-full-ai-suite-after-codex-cli-debut.aspx#:~:text=GPT,HD%20synthetic%20voices%20from%20text). Codex CLI sollte auch diese Modelle ansprechen können, sofern man den korrekten Modellnamen hat (z.B. gpt-4 oder gpt-4-32k im Rahmen der API). Praktisch wird man aber eher die neueren GPT-4.1-Modelle oder die o3/o4-Minis nutzen, da OpenAI diese optimiert hat.

**Azure-Modelle:** Die Dokumentation fragt explizit nach Azure-OpenAI. **Ja, Codex CLI unterstützt Azure OpenAI-Instanzen**, jedoch erfordert dies etwas Konfiguration. Da Azure-eigene Endpunkte und API-Versionsparameter genutzt werden, muss man in der ~/.codex/config.toml einen neuen Modell-Provider definieren. Das offizielle README zeigt ein Beispiel für Azure[[26]](https://github.com/openai/codex#:~:text=,appended%20to%20the%20request%20URL)[[27]](https://github.com/openai/codex#:~:text=Once%20you%20have%20defined%20a,your%20default%20provider%20as%20follows). Man legt einen Provider-Eintrag an, z.B.:

[model\_providers.azure]  
name = "Azure"  
base\_url = "https://<YOUR\_RESOURCE\_NAME>.openai.azure.com/openai"  
env\_key = "AZURE\_OPENAI\_API\_KEY" # oder OPENAI\_API\_KEY, je nachdem  
query\_params = { api-version = "2025-04-01-preview" }  
wire\_api = "responses"

Wichtig ist hier wire\_api = "responses", damit das CLI weiß, dass dieser Provider das Chat-Completions-Protokoll (“Responses API”) spricht[[26]](https://github.com/openai/codex#:~:text=,appended%20to%20the%20request%20URL). Ebenso muss der Pfad stimmen (Azure erwartet meist /openai/deployments/<Name>/... – durch base\_url und ggf. Angabe des Deploymentnamens als Modell). **Die Modellbezeichnung im CLI-Aufruf entspricht dann dem** Deployment-Namen **in Azure!** Beispielsweise, hat man in Azure ein Deployment "gpt-35" für GPT-3.5-Turbo eingerichtet, würde man codex -m gpt-35 --config model\_provider=azure verwenden. Alternativ kann man in der Config ein Profil hinterlegen, z.B.:

model\_provider = "azure"  
model = "<DeploymentName>"

im [profiles.<name>], um mit codex --profile <name> schnell darauf zuzugreifen[[28]](https://github.com/openai/codex#:~:text=model_provider%20%3D%20). Zusammengefasst: **Azure-unterstützung ist möglich**, aber nicht “out of the box” – man muss die Konfiguration um den eigenen Azure-Endpunkt (inkl. API-Key als Environment-Variable) ergänzen. Die Doku des Fragestellers erwähnt das kurz („Azure erfordert passenden Provider und Deployment-Namen als model“), was **korrekt** ist. Offizielle Quellen bestätigen dieses Vorgehen (siehe oben). Dieser Teil der Dokumentation ist also **richtig**, lediglich sehr knapp gehalten. Ein möglicher Hinweis zur Erweiterung: man könnte die Beispiel-TOML wie oben in die Doku aufnehmen, damit Nutzer wissen, wie Azure einzurichten ist.

**Eigene Deployments / andere API-Hosts:** Ähnlich wie Azure kann Codex CLI auch **beliebige andere Endpunkte** ansprechen, **solange sie OpenAI-kompatible APIs** bieten. Im README werden z.B. **OpenRouter** und **OpenAI-kompatible OSS-Server** genannt[[29]](https://github.com/openai/codex#:~:text=%5Bmodel_providers.openrouter%5D%20name%20%3D%20,OPENROUTER_API_KEY)[[30]](https://github.com/openai/codex#:~:text=For%20providers%20that%20also%20require,when%20making%20a%20request). Man kann also in model\_providers z.B. einen Eintrag für OpenRouter (oder eine selbst gehostete API) definieren und via base\_url und ggf. env\_key (für Auth-Token) konfigurieren. Das CLI ist sehr flexibel: es unterstützt sowohl OpenAI’s ursprüngliches Completion-Protokoll als auch das neuere Chat-Completion-Format („Responses API“) – letzteres ist Standard. Für eigene Deployments (z.B. wenn man **ein eigenes Modell fine-tuned und über einen kompatiblen Endpoint anbietet**) gilt: Wenn es das /v1/chat/completions Format spricht, kann Codex darauf zugreifen. Im Zweifel kann man wire\_api="responses" setzen, um das zu erzwingen. Die Frage nennt „Responses API“ – damit ist vermutlich genau diese Kompatibilität gemeint. **Ergebnis:** Die bestehende Dokumentation ist hier **nicht falsch**, aber sie könnte genauer sein: Außer *o3*, *o4-mini* und *gpt-5* sind **alle OpenAI-Modelle von GPT-3.5 bis 4.1 (auch Turbo- und 32k-Varianten)** verwendbar, ebenso eventuell neu erscheinende (Codex CLI wird mit Updates angepasst). Sogar völlig andere Anbieter (Azure, OpenRouter, lokale Server) können eingebunden werden, was ein großer Pluspunkt ist. Dies sollte in der Dokumentation ggf. ausführlicher erklärt werden, z.B. mit einem Abschnitt *“Modelle und Provider wechseln”* inklusive Beispiel für Azure.

Ein potenziell **veralteter Hinweis** in der Doku ist die Nennung von *“gpt-4.1-mini”* – dieser Begriff stammt wohl aus dem Zeitraum direkt nach Einführung von GPT-4.1. Offiziell würde man das Modell entweder *o4-mini* (alte Bezeichnung) oder *GPT-4.1 Mini* nennen. Da OpenAI die *o*-Präfixe teilweise abgelöst hat, könnte man die Doku hier aktualisieren: GPT-4.1 (und dessen Derivate Mini/Nano) sind die korrekten Bezeichnungen. Ebenso könnte man klarstellen, dass “gpt-5” im Kontext Codex CLI kein eigenständiges API-Modell, sondern die interne Auswahl des besten verfügbaren Modells ist (für Plus-User). Einige Nutzer waren verwirrt und dachten, GPT-5 sei offiziell veröffentlicht – das ist **nicht der Fall**, es handelt sich eher um eine **Marketing/Platzhalter-Bezeichnung** innerhalb der ChatGPT-Integration[[18]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Which%20models%20does%20Codex%20use%3F). Ggf. sollte die Doku erwähnen: *„Standardmodell (bei ChatGPT-Login) ist derzeit* *GPT-5 (Alias für GPT-4.1-derivates Modell).“* und für API-Key-Nutzung: *„Standardmodell per API ist o4-mini, falls nicht umgestellt.“*

## Nutzung lokaler Open-Source-Modelle mit Codex CLI (Ollama und GPT-OSS)

Ein besonders interessanter Punkt ist die **Integration von Open-Source-LLMs** in Codex CLI. Die Dokumentation nennt als Beispiel die Verwendung über **Ollama** mit Modellen *gpt-oss:20b* und *gpt-oss:120b*. Dies ist **aktueller Stand und offiziell unterstützt**. OpenAI hat im Jahr 2025 zwei eigene open-weight Modelle veröffentlicht, **GPT-OSS-20B und GPT-OSS-120B**, die frei unter Apache 2.0 Lizenz verfügbar sind[[31]](https://openai.com/index/gpt-oss-model-card/#:~:text=gpt,oss%20usage%20policy)[[32]](https://www.lesswrong.com/posts/oPqgSrfkTt2sCKM3e/openai-releases-gpt-oss#:~:text=OpenAI%20releases%20gpt,world%20performance%20at). Codex CLI bietet dafür den **--oss-Modus**, der es dem Agent ermöglicht, *vollständig lokal* auf diese (oder andere) Modelle zuzugreifen.

**Funktionsweise:** Hinter --oss steckt defaultmäßig ein vordefinierter Provider namens **“oss”**, der auf einen lokalen Server horcht. Standard ist die Integration mit **Ollama** (ein lokaler LLM-Server). Im README ist beschrieben, dass Codex mit --oss auf http://localhost:11434/v1 zugreift[[33]](https://github.com/openai/codex#:~:text=Model%20selection%20when%20using%20%60)[[34]](https://github.com/openai/codex#:~:text=%2A%20By%20default%2C%20%60,oss) – genau der Standard-Port von Ollama. Wenn man also **Ollama installiert und das GPT-OSS-Modell lädt**, kann Codex CLI darauf zugreifen. Die Doku des Fragestellers sagt richtig: *“Basis-URL per CODEX\_OSS\_BASE\_URL/CODEX\_OSS\_PORT oder config.toml setzen”* – per Umgebungsvariable kann man einen anderen Host/Port angeben, falls Ollama z.B. auf einem anderen Port oder Remote läuft[[34]](https://github.com/openai/codex#:~:text=%2A%20By%20default%2C%20%60,oss). Man kann auch in der Config den Provider “oss” überschreiben, z.B. um dauerhaft base\_url = "http://myserver:11434/v1" zu setzen[[35]](https://github.com/openai/codex#:~:text=by%20overriding%20the%20built,codex%2Fconfig.toml).

**Modellauswahl bei --oss:** Wird kein Modell explizit angegeben, nutzt Codex **standardmäßig -m gpt-oss:20b**[[33]](https://github.com/openai/codex#:~:text=Model%20selection%20when%20using%20%60). Dieser Name ist in Ollama als Alias konfiguriert, um das 20-Milliarden-Modell zu laden. Es wird sogar **automatisch geprüft und heruntergeladen**, falls es noch nicht lokal vorhanden ist[[36]](https://www.reddit.com/r/LocalLLaMA/comments/1miq7sp/gptoss_support_merged_into_codex/#:~:text=GPT,pick%20a%20different%20size%2C)[[33]](https://github.com/openai/codex#:~:text=Model%20selection%20when%20using%20%60) – sprich Codex CLI triggert Ollama, das Modell nachzuladen. Möchte man das größere 120B-Modell nutzen, kann man -m gpt-oss:120b angeben[[37]](https://github.com/openai/codex#:~:text=it%20exists%20locally%20,oss%3A120b). Alternativ lassen sich natürlich auch **andere Modelle** verwenden, die Ollama bereitstellt. Zum Beispiel zeigt das README ein Profil für *Mistral* (ein 7B Open-Source-Modell):

[profiles.mistral]  
model\_provider = "ollama"  
model = "mistral"

Mit codex --profile mistral --oss würde man dann dieses Modell nutzen[[38]](https://github.com/openai/codex#:~:text=configuration%20like%20so%3A). Generell ist Codex CLI also nicht auf GPT-OSS beschränkt – **jedes Modell, das Ollama verwalten kann und über dessen API abrufbar ist, kann eingebunden werden**. GPT-OSS-20B/120B sind lediglich die von OpenAI empfohlenen Open-Source-Pendants, da sie für Reasoning-Aufgaben optimiert wurden und der o3/o4-Performance nahekommen[[39]](https://openai.com/index/introducing-gpt-oss/#:~:text=The%20gpt,16%20GB%20of%20memory%2C)[[40]](https://www.infoq.com/news/2025/08/openai-gpt-oss/#:~:text=The%20gpt,without%20reliance%20on%20cloud%20infrastructure).

**Kompatibilität:** Codex CLI schickt an den --oss-Provider dieselben Anfragen wie an die OpenAI-API (Chat-Completion). Ollama v0.0.13+ unterstützt dieses Chat-Endpunkt-Format, sodass die Integration nahtlos funktioniert. In der Praxis bedeutet das: **Multimodalität ist lokal eingeschränkt.** Die GPT-OSS-Modelle sind reine Sprachmodelle (mit großem Kontext, aber **ohne** native Bildverarbeitung). Funktionen wie das Interpretieren von Screenshots oder Diagrammen, die Codex mit OpenAI-Modellen beherrscht[[41]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=,upgrade%60%29%20gets%20you%20started), stehen lokal nur zur Verfügung, wenn das Modell entsprechend trainiert wurde (was bei GPT-OSS nicht der Fall ist). Für typische Coding-Aufgaben ist das aber nebensächlich. Ansonsten unterstützt GPT-OSS *Chain-of-thought Reasoning* und das Ausgeben von überlegten Zwischenschritten ähnlich wie die geschlossenen Modelle[[42]](https://www.infoq.com/news/2025/08/openai-gpt-oss/#:~:text=infrastructure)[[43]](https://www.infoq.com/news/2025/08/openai-gpt-oss/#:~:text=Trained%20using%20techniques%20adapted%20from,4o). Die Codex-typische Sandbox zum Ausführen von Befehlen wird natürlich weiterhin genutzt – lokal darf der Agent ebenfalls Shell-Kommandos ausführen (mit den gleichen Beschränkungen), nur kommen die KI-Antworten eben von einem anderen Modell.

**Leistung und Einschränkungen:** Hier gibt es einige praktische Punkte: Das **20B-Modell** ist darauf optimiert, mit ca. 16 GB RAM lauffähig zu sein (z.B. auf einer starken CPU oder Apple Silicon)[[44]](https://www.infoq.com/news/2025/08/openai-gpt-oss/#:~:text=mixture,without%20reliance%20on%20cloud%20infrastructure). Es nutzt ein Mixture-of-Experts-Architektur, wobei jeweils nur ~3.6 Mrd. Parameter pro Token aktiv sind[[40]](https://www.infoq.com/news/2025/08/openai-gpt-oss/#:~:text=The%20gpt,without%20reliance%20on%20cloud%20infrastructure). Die **120B-Variante** ist weitaus größer – benötigt ~80 GB GPU-RAM für performante Nutzung[[40]](https://www.infoq.com/news/2025/08/openai-gpt-oss/#:~:text=The%20gpt,without%20reliance%20on%20cloud%20infrastructure). Diese Modelle können also deutlich langsamer sein als die Cloud-Modelle und erfordern entsprechende Hardware (vor allem das 120B). Für viele Nutzer wird *gpt-oss-20b* der praktikablere Kompromiss sein. In der Doku könnte man dies als Hinweis aufnehmen, z.B.: *„Das 120B-Modell ist nur auf High-End-GPUs sinnvoll nutzbar; für Alltagsgeräte empfiehlt sich 20B.“*. Außerdem behalten die **Open-Source-Modelle nicht unbedingt den gleichen Grad an “Vorsicht”** ein wie OpenAIs gehostete Modelle – z.B. könnten sie anfälliger für falsche Codeänderungen sein, da das RLHF/Feintuning anders gelagert ist (OpenAI hat allerdings die GPT-OSS-Modelle gezielt auf Code, Mathe, etc. getestet[[45]](https://www.infoq.com/news/2025/08/openai-gpt-oss/#:~:text=Trained%20using%20techniques%20adapted%20from,4o) und Sicherheitsmechanismen geprüft). Ein weiterer Punkt: **Zero Data Retention** – bei lokalen Modellen stellt sich die Frage nicht, da nichts an OpenAI geschickt wird. In der Cloud kann man ZDR mittels disable\_response\_storage=true aktivieren[[46]](https://github.com/openai/codex#:~:text=Zero%20data%20retention%20)[[47]](https://github.com/openai/codex#:~:text=Ensure%20you%20are%20running%20,command%20line%20option%20each%20time), wie die Doku auch erwähnte. Für Vollständigkeit: in lokaler Nutzung ist man komplett offline.

Die Dokumentation des Fragestellers fasst die **Integration korrekt zusammen**: codex --oss -m gpt-oss:20b genügt im Prinzip (vorausgesetzt, Ollama ist installiert und läuft). Die Erwähnung der Konfigurierbarkeit via config.toml ist ebenfalls korrekt und wird im offiziellen README bestätigt[[48]](https://github.com/openai/codex#:~:text=by%20overriding%20the%20built,codex%2Fconfig.toml). Einschränkungen wurden vermutlich nicht detailliert diskutiert, hier wäre eine **Empfehlung für die Doku**, die oben genannten Punkte aufzunehmen: Hardware-Anforderungen, kein Bildverständnis, ggf. geringere Geschwindigkeit. Ansonsten ist dies ein sehr aktuelles Feature – OpenAI hat GPT-OSS erst im August 2025 veröffentlicht[[49]](https://www.infoq.com/news/2025/08/openai-gpt-oss/#:~:text=OpenAI%20has%20released%20gpt,0%20license), die vorhandene Dokumentation scheint diese Neuerung bereits erfasst zu haben (inkl. Verweis auf Reddit/Community für Tipps, was sinnvoll ist – etwa die Config statt jedes Mal --config Flags zu nutzen, wie in der Doku erwähnt).

## Offiziell unterstützte erweiterte Optionen und aktuelle Entwicklung

Zusammenfassend hier eine **Liste der wichtigsten offiziell unterstützen Optionen/Funktionen**, teils schon angerissen, zur Validierung der Dokumentation:

* **Profile in config.toml:** *Offiziell unterstützt.* Man kann in ~/.codex/config.toml Profile definieren (z.B. verschiedene Voreinstellungen für Full-Auto vs Read-Only, oder Cloud vs Local Modelle)[[50][51]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=). Dies wird weiterhin gepflegt. Die Dokumentation zeigt Beispiele ([profiles.full\_auto], [profiles.readonly\_quiet] etc.), was korrekt ist. Die Syntax hat sich nicht geändert.
* **Sandbox-Einstellungen:** *Offiziell unterstützt.* Werte wie sandbox\_mode = "workspace-write" oder "read-only" sind nach wie vor gültig[[52]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=sandbox_mode%20%20%20%20%3D,im%20Workspace%20schreiben%2C%20sonst%20fragen)[[53]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=). Zusätzlich kann man pro Sandbox-Modus Sub-Konfiguration setzen (z.B. [sandbox\_workspace\_write] network\_access = false wie in der Doku genannt, um ggf. Netzwerk zu erlauben – default ist false)[[54]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=). Dieser Teil der Doku ist korrekt; die Defaults (kein Netzwerk) stimmen mit GitHub überein.
* **CLI-Flags:** Alle wichtigen Flags, die in der Dokumentation aufgeführt sind, sind aktuell:
* --auto-edit, --full-auto (siehe Modi oben)[[55]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=4,as%20needed),
* --sandbox <mode> und --ask-for-approval <policy> (die Doku nannte z.B. on-request, never – das sind gültige Werte)[[14]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=codex%20),
* --model/-m <name> zur Modellwahl[[18]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Which%20models%20does%20Codex%20use%3F),
* --login (für ChatGPT OAuth Login) und auch --free (die Doku erwähnt Promo-Credits via codex --free). Hierzu: *OpenAI hatte zeitweise ein Kontingent an kostenlosen Test-Credits für Codex CLI bereitgestellt*. codex --free öffnet einen Browser-Link, um Promo-Credits zu aktivieren, wenn man keinen Plus-Account hat. Dies ist kein Kernfeature, eher eine temporäre Aktion – aber noch gültig für neue Nutzer (Stand einiger Forenposts vom Launch). In der offiziellen Hilfe wird --free nicht prominent erwähnt, aber es existierte als Option. Es wäre sinnvoll zu prüfen, ob diese Option aktuell noch angeboten wird; sollte sie entfallen sein, wäre die Doku hier zu korrigieren. Stand **August 2025** allerdings gibt es Berichte, dass neue User 50$ API Credits für Codex CLI erhalten können – somit dürfte codex --free noch funktionieren (dies ist jedoch **inoffizielle Info**, ggf. vorsichtig behandeln).
* Weitere Flags: --exec "<prompt>" (nicht interaktiver Modus) wurde in der Doku als codex exec --full-auto "Anweisung" gezeigt[[56]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=codex%20exec%20,for%20next%20release) – das ist korrekt (dieser Modus für CI ist unterstützt und unverändert)[[57]](https://github.com/openai/codex#:~:text=Non).
* --upgrade zum Aktualisieren existiert ebenfalls (einfach npm update oder brew upgrade intern)[[58]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=How%20do%20I%20update%20Codex%3F).
* --version gibt die CLI-Version – auch das erwähnt die Doku (zur Prüfung der Installation)[[59]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=npm%20install%20,CLI%20global%20installieren).

Insgesamt stimmen die aufgezählten CLI-Befehle und Optionen weitgehend mit der offiziellen Referenz überein. Man sollte die Doku lediglich auf dem neuesten Stand halten, was Default-Werte angeht (z.B. Default-Modell). Ein Blick in codex --help oder das README der jeweiligen Version ist empfehlenswert, da sich Kleinigkeiten ändern können.

* **MCP (Model Context Protocol):** Die Doku streift es (MCP-Server einbinden). Dies ist ein sehr avanciertes Feature, das in der offiziellen README als experimentell erwähnt wird[[60]](https://github.com/openai/codex#:~:text=Model%20Context%20Protocol%20). Nur der Vollständigkeit halber: MCP erlaubt es, externe Tools einzubinden, die speziellen Kontext liefern – für die meisten Nutzer unwichtig. Der Doku-Ausschnitt dazu ist nicht falsch, aber vermutlich über 99% der Anwendungsfälle nicht relevant. Das Feature ist noch im Fluss.
* **Plattformen:** Offiziell unterstützt Codex CLI Linux und macOS; Windows nur via WSL2[[61]](https://github.com/openai/codex#:~:text=Does%20it%20work%20on%20Windows%3F). Die Dokumentation sagt sinngemäß das Gleiche (WSL2 empfohlen, Sandbox auf nativem Windows nicht verfügbar)[[62]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=Windows). Das ist weiterhin aktuell und wichtig (auch im Help Center steht, Windows-Unterstützung sei experimentell)[[63]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=What%20platforms%20are%20supported%3F). Also alles korrekt.

**Empfohlene Dokumentations-Updates:**  
Anhand der obigen Punkte würde ich folgende **Korrekturen/Ergänzungen** vorschlagen:

* **Rollen/Agents.md Abschnitt präzisieren:** Klarstellen, dass Core/Advanced/Specialized nur beispielhafte Kategorien sind. Evtl. darauf hinweisen, dass Codex CLI standardmäßig einen Agenten hat und AGENTS.md als Anleitung nutzt[[1]](https://github.com/openai/codex#:~:text=Memory%20%26%20project%20docs). Vorschlag: *„Codex CLI kann durch eine AGENTS.md-Datei mit Projektwissen versorgt werden. Die dort definierten Rollen (z.B. ‘Executor’, ‘Validator’…) dienen zur Strukturierung der KI-Vorgehensweise, sind jedoch keine separaten KI-Module – alle Aufgaben werden vom Codex-Modell im Rahmen einer Session erledigt.“* Zudem könnte man erwähnen, dass parallele Subagenten à la Claude nicht vorhanden sind, um Erwartungen zu managen.
* **Modell-Auswahl aktualisieren:** Die Doku sollte die **Unterstützung aller Modelle** betonen. Beispielsweise anstatt nur “gpt-4.1-mini” zu nennen, lieber *„Modelle: Standard gpt-5 (bei Plus-Nutzung) entsprechend neuester GPT-4.1-Generation. Alternativ via -m z.B. o3 (GPT-3.5 Turbo), gpt-4.1 (großes GPT-4.1-Modell), gpt-4 (Original GPT-4)… sowie zukünftige Modelle.“* Ein Hinweis auf **Azure** mit kurzem Config-Snippet wäre sehr hilfreich für Leser, die das nutzen wollen[[64]](https://github.com/openai/codex#:~:text=%5Bmodel_providers.azure%5D%20name%20%3D%20%22Azure%22%20,preview%22%20%7D%20wire_api%20%3D%20%22responses)[[28]](https://github.com/openai/codex#:~:text=model_provider%20%3D%20).
* **Lokale Modelle:** Hier könnte man ergänzen, dass OpenAI’s GPT-OSS Modelle eingebunden werden können und wie Ollama installiert/genutzt wird (vielleicht einen Satz dazu, dass man Ollama installieren muss, bevor --oss funktioniert). Außerdem die **Hardwareanforderungen** und der **Scope** (kein Multimodal) der lokalen Modelle, damit Nutzer wissen, was sie erwartet. Die vorhandene Doku ist diesbezüglich schon recht informativ (nennt z.B. „für Air-gapped Betrieb config.toml statt Flags nutzen – etablierter Community-Tipp“, was aus einem Reddit-Tipp stammt und valide ist). Ggf. könnte man noch auf die **Performance-Unterschiede** hinweisen: *„Lokale Modelle sind spürbar langsamer und weniger leistungsfähig als die OpenAI-Cloud-Modelle (insb. GPT-OSS-20B vs GPT-4.1), eignen sich aber für Offline-Nutzung oder als Fallback.“* Auch die Info, dass GPT-OSS **128k Token Kontext** unterstützt[[45]](https://www.infoq.com/news/2025/08/openai-gpt-oss/#:~:text=Trained%20using%20techniques%20adapted%20from,4o), könnte interessant sein, falls das Anwendungsfälle eröffnet, die man mit Standard-3.5/4 nicht ohne Weiteres abdecken konnte.
* **Sonstige Fehler/Veraltetheiten:** Insgesamt scheint die bestehende Dokumentation viele richtige Informationen zu enthalten (Installationsanleitung, Nutzung, Optionen – alles stimmig). Einige Begriffe könnten modernisiert werden (z.B. „o4-mini“ statt „gpt-4.1-mini“ oder eben deutlicher erklärt werden). Auch sollte man, wie oben erwähnt, „GPT-5“ erklären, da dies ohne Kontext irritierend wirkt.
* **Quellen & Aktualität:** Da Codex CLI sehr aktiv weiterentwickelt wird (der GitHub hat im Moment hunderte Commits und häufige Releases), sollte die Doku lebende Links zur offiziellen README bzw. Help Center enthalten, um Details nachzuschlagen. Zum Beispiel: *„(Stand August 2025, Codex CLI v0.21)“* könnte man anführen, und empfehlen, codex --help für die neueste Liste der Flags zu konsultieren[[65]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=Hinweis%3A%20Die%20vollst%C3%A4ndige%20Liste%20ist,%28GitHub). Die Doku hatte das sinngemäß erwähnt („Liste ist im Fluss, Releases konsultieren“) – dem ist weiterhin zuzustimmen.

Zusammengefasst sind **keine gravierenden Fehler** in der Dokumentation auszumachen – hauptsächlich handelt es sich um **Nuancierungen und Klarstellungen**: Die Rollenkonzepte sind **nicht offiziell getrennt**, sondern via AGENTS.md umzusetzen; es lassen sich **weit mehr Modelle** nutzen als vielleicht anfänglich gedacht (inkl. Azure und künftige); und die **lokale Modellnutzung** ist ein voll unterstütztes Feature, das mit den GPT-OSS-Modellen aktuell sehr spannend ist. Mit den vorgeschlagenen Ergänzungen wäre die Dokumentation auf dem neuesten Stand und für Anwender noch hilfreicher.

**Quellen:** Die obigen Aussagen wurden mit der offiziellen OpenAI-README[[66]](https://github.com/openai/codex#:~:text=Which%20models%20are%20supported%3F)[[33]](https://github.com/openai/codex#:~:text=Model%20selection%20when%20using%20%60), dem Help Center[[18]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Which%20models%20does%20Codex%20use%3F) sowie Entwickler-Community-Beiträgen[[67]](https://www.reddit.com/r/ChatGPTCoding/comments/1m5uloy/are_there_any_real_benefits_in_using_terminalcli/#:~:text=Actually%2C%20Claude%20Code%20is%20much,secrete%20sauce%20of%20Claude%20Code)[[31]](https://openai.com/index/gpt-oss-model-card/#:~:text=gpt,oss%20usage%20policy) abgeglichen, um sicherzustellen, dass der Stand *August 2025* korrekt wiedergegeben ist.

[[1]](https://github.com/openai/codex#:~:text=Memory%20%26%20project%20docs) [[15]](https://github.com/openai/codex#:~:text=The%20mechanism%20Codex%20uses%20to,policy%20depends%20on%20your%20OS) [[17]](https://github.com/openai/codex#:~:text=Which%20models%20are%20supported%3F) [[23]](https://github.com/openai/codex#:~:text=Run%20,%28Enterprise%20is%20coming%20soon) [[26]](https://github.com/openai/codex#:~:text=,appended%20to%20the%20request%20URL) [[27]](https://github.com/openai/codex#:~:text=Once%20you%20have%20defined%20a,your%20default%20provider%20as%20follows) [[28]](https://github.com/openai/codex#:~:text=model_provider%20%3D%20) [[29]](https://github.com/openai/codex#:~:text=%5Bmodel_providers.openrouter%5D%20name%20%3D%20,OPENROUTER_API_KEY) [[30]](https://github.com/openai/codex#:~:text=For%20providers%20that%20also%20require,when%20making%20a%20request) [[33]](https://github.com/openai/codex#:~:text=Model%20selection%20when%20using%20%60) [[34]](https://github.com/openai/codex#:~:text=%2A%20By%20default%2C%20%60,oss) [[35]](https://github.com/openai/codex#:~:text=by%20overriding%20the%20built,codex%2Fconfig.toml) [[37]](https://github.com/openai/codex#:~:text=it%20exists%20locally%20,oss%3A120b) [[38]](https://github.com/openai/codex#:~:text=configuration%20like%20so%3A) [[46]](https://github.com/openai/codex#:~:text=Zero%20data%20retention%20) [[47]](https://github.com/openai/codex#:~:text=Ensure%20you%20are%20running%20,command%20line%20option%20each%20time) [[48]](https://github.com/openai/codex#:~:text=by%20overriding%20the%20built,codex%2Fconfig.toml) [[57]](https://github.com/openai/codex#:~:text=Non) [[60]](https://github.com/openai/codex#:~:text=Model%20Context%20Protocol%20) [[61]](https://github.com/openai/codex#:~:text=Does%20it%20work%20on%20Windows%3F) [[64]](https://github.com/openai/codex#:~:text=%5Bmodel_providers.azure%5D%20name%20%3D%20%22Azure%22%20,preview%22%20%7D%20wire_api%20%3D%20%22responses) [[66]](https://github.com/openai/codex#:~:text=Which%20models%20are%20supported%3F) GitHub - openai/codex: Lightweight coding agent that runs in your terminal

<https://github.com/openai/codex>

[[2]](https://www.reddit.com/r/ChatGPTCoding/comments/1m5uloy/are_there_any_real_benefits_in_using_terminalcli/#:~:text=Actually%2C%20Claude%20Code%20is%20much,secrete%20sauce%20of%20Claude%20Code) [[3]](https://www.reddit.com/r/ChatGPTCoding/comments/1m5uloy/are_there_any_real_benefits_in_using_terminalcli/#:~:text=Claude%20Code%20extensively%20uses%20subagents,search%20of%20Cline%2F%20RooCode%2F%20ZentaraCode) [[67]](https://www.reddit.com/r/ChatGPTCoding/comments/1m5uloy/are_there_any_real_benefits_in_using_terminalcli/#:~:text=Actually%2C%20Claude%20Code%20is%20much,secrete%20sauce%20of%20Claude%20Code) Are there any real benefits in using terminal/CLI agents instead of those inside code editor? : r/ChatGPTCoding

<https://www.reddit.com/r/ChatGPTCoding/comments/1m5uloy/are_there_any_real_benefits_in_using_terminalcli/>

[[4]](https://agentsmd.net/#:~:text=Agents,project%27s%20structure%2C%20conventions%2C%20and%20requirements) [[5]](https://agentsmd.net/#:~:text=2) Agents.md Guide for OpenAI Codex - Enhance AI Coding

<https://agentsmd.net/>

[[6]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Approval%20Modes) [[7]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Full%20Auto) [[8]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Suggest%20) [[9]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Auto%20Edit) [[10]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Full%20Auto) [[11]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=How%20do%20I%20change%20approval,modes) [[12]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Read%2C%20write%2C%20and%20execute%20commands,scoped%20to%20the%20current%20directory) [[13]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Read%2C%20write%2C%20and%20execute%20commands,scoped%20to%20the%20current%20directory) [[18]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=Which%20models%20does%20Codex%20use%3F) [[41]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=,upgrade%60%29%20gets%20you%20started) [[55]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=4,as%20needed) [[58]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=How%20do%20I%20update%20Codex%3F) [[63]](https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started#:~:text=What%20platforms%20are%20supported%3F) OpenAI Codex CLI – Getting Started | OpenAI Help Center

<https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started>

[[14]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=codex%20) [[50]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=) [[51]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=) [[52]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=sandbox_mode%20%20%20%20%3D,im%20Workspace%20schreiben%2C%20sonst%20fragen) [[53]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=) [[54]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=) [[56]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=codex%20exec%20,for%20next%20release) [[59]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=npm%20install%20,CLI%20global%20installieren) [[62]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=Windows) [[65]](file://file-P9gZhYXperJ4D9d1AQfNig#:~:text=Hinweis%3A%20Die%20vollst%C3%A4ndige%20Liste%20ist,%28GitHub) Installations und Verwendungsdokumentation Codex CLI.docx

<file://file-P9gZhYXperJ4D9d1AQfNig>

[[16]](https://github.com/openai/codex/issues/2182#:~:text=,%7D%20%7D%20%5Bmodel_providers) Bug: Codex TUI does not render assistant messages for Azure Chat ...

<https://github.com/openai/codex/issues/2182>

[[19]](https://visualstudiomagazine.com/articles/2025/04/17/mapping-openais-full-ai-suite-after-codex-cli-debut.aspx#:~:text=%2A%20New%20,to%20explore%2C%20use%2C%20and%20extend) [[20]](https://visualstudiomagazine.com/articles/2025/04/17/mapping-openais-full-ai-suite-after-codex-cli-debut.aspx#:~:text=execution%20Secure%20coding%2C%20automation%2C%20terminal,4o%20Code%20editing%2C%20AI) [[21]](https://visualstudiomagazine.com/articles/2025/04/17/mapping-openais-full-ai-suite-after-codex-cli-debut.aspx#:~:text=On%20the%20same%20day%20that,Mini%2C%20adding%20to%20the%20mix) [[22]](https://visualstudiomagazine.com/articles/2025/04/17/mapping-openais-full-ai-suite-after-codex-cli-debut.aspx#:~:text=workflows%20in%20constrained%20environments%20GPT,cost%20queries) [[24]](https://visualstudiomagazine.com/articles/2025/04/17/mapping-openais-full-ai-suite-after-codex-cli-debut.aspx#:~:text=April%2014%2C%202025%201M,char%20limit%20Audiobooks) [[25]](https://visualstudiomagazine.com/articles/2025/04/17/mapping-openais-full-ai-suite-after-codex-cli-debut.aspx#:~:text=GPT,HD%20synthetic%20voices%20from%20text) Mapping OpenAI's Full AI Suite After Codex CLI Debut -- Visual Studio Magazine

<https://visualstudiomagazine.com/articles/2025/04/17/mapping-openais-full-ai-suite-after-codex-cli-debut.aspx>

[[31]](https://openai.com/index/gpt-oss-model-card/#:~:text=gpt,oss%20usage%20policy) gpt-oss-120b & gpt-oss-20b Model Card | OpenAI

<https://openai.com/index/gpt-oss-model-card/>

[[32]](https://www.lesswrong.com/posts/oPqgSrfkTt2sCKM3e/openai-releases-gpt-oss#:~:text=OpenAI%20releases%20gpt,world%20performance%20at) OpenAI releases gpt-oss - LessWrong

<https://www.lesswrong.com/posts/oPqgSrfkTt2sCKM3e/openai-releases-gpt-oss>

[[36]](https://www.reddit.com/r/LocalLLaMA/comments/1miq7sp/gptoss_support_merged_into_codex/#:~:text=GPT,pick%20a%20different%20size%2C) GPT-OSS Support Merged into Codex : r/LocalLLaMA - Reddit

<https://www.reddit.com/r/LocalLLaMA/comments/1miq7sp/gptoss_support_merged_into_codex/>

[[39]](https://openai.com/index/introducing-gpt-oss/#:~:text=The%20gpt,16%20GB%20of%20memory%2C) Introducing gpt-oss - OpenAI

<https://openai.com/index/introducing-gpt-oss/>

[[40]](https://www.infoq.com/news/2025/08/openai-gpt-oss/#:~:text=The%20gpt,without%20reliance%20on%20cloud%20infrastructure) [[42]](https://www.infoq.com/news/2025/08/openai-gpt-oss/#:~:text=infrastructure) [[43]](https://www.infoq.com/news/2025/08/openai-gpt-oss/#:~:text=Trained%20using%20techniques%20adapted%20from,4o) [[44]](https://www.infoq.com/news/2025/08/openai-gpt-oss/#:~:text=mixture,without%20reliance%20on%20cloud%20infrastructure) [[45]](https://www.infoq.com/news/2025/08/openai-gpt-oss/#:~:text=Trained%20using%20techniques%20adapted%20from,4o) [[49]](https://www.infoq.com/news/2025/08/openai-gpt-oss/#:~:text=OpenAI%20has%20released%20gpt,0%20license) OpenAI Releases gpt-oss-120b and gpt-oss-20b, Open-Weight Language Models for Local Deployment - InfoQ

<https://www.infoq.com/news/2025/08/openai-gpt-oss/>