Codex

实用技巧与工具
Codex App/CLI 使用技巧、 Provider 切换、历史记录管理、工具推荐等实用指南 Codex 切换 Provider 保留历史记录全方案（ 2026.06）
如何与 Codex 协作？

Codex 切换 Provider 保留历史记录全方案（2026.06）

> 数据来源: LINUX DO 社区 20+ 帖子逐楼阅读, 2026-06-08 整理。覆盖 Codex App/CLI 切换 Provider 后历史会话丢失的所有解决方案。

一、问题本质

Codex 按 **model_provider** 维度隔离会话。切换 Provider 后，历史会话并没有消失，只是在当前 Provider 空间里看不到索引/元数据了。

涉及的本地文件：

- **~/.codex/config.toml** — Provider 配置
- **~/.codex/state_5.sqlite** — 线程的 Provider 信息
- **~/.codex/sessions/**/rollout-*.jsonl** — 会话内容（首行有 Provider 元数据）
- **~/.codex/session_index.jsonl** — 会话索引

> 来源: 2139733「Codex + CC Switch 切换供应商后 codex会话消失/401 的完整修法」

二、解决方案总览

方案	名称	难度	耗时	适用场景
方案 1	统一 Provider 名称	□	10秒	预防为主，配置时统一命名
方案 2	修改 SQLite 数据库	□□	1分钟	已切换，需要恢复历史

方案	名称	难度	耗时	适用场景
方案3	codex-provider-sync工具	□	30秒	一键恢复，推荐新手
方案4	Codex-Threadripper工具	□□	持续运行	经常切换Provider的用户
方案5	CLI直接resume	□	即时	只恢复单个会话

三、方案详解

方案1：统一Provider名称（最简单）

核心思路：让所有中转站使用同一个Provider名称，这样Codex就不会隔离会话。

操作步骤：

1. 打开CC Switch或中转站配置
2. 把Provider名称改成和之前一样（比如都叫 `custom`）
3. 重启Codex App

config.toml 示例：

```
model_provider = "custom"
[model_providers.custom]
name = "custom"
base_url = "http://your-proxy/v1"
```

⚠ 注意：不能用 `openai`（小写）作为Provider名称，这是官方保留字，会报错。

> “来源：[2098635#1](#) @hwj1「分享一个10秒钟解决ccswitch切换中转站导致codex历史会话消失的问题」

方案2：修改SQLite数据库

操作步骤：

1. 关闭Codex App
2. 用数据库工具打开 `~/.codex/state_5.sqlite`
3. 修改 `threads` 表的 `model_provider` 字段，改成你当前的Provider名称
4. 重启Codex App

SQL 命令：

```
UPDATE threads SET model_provider = 'custom';
```

📖 提示：如果想享受远程压缩功能，可以把所有会话的 Provider 都改成 `OpenAI`（大写）。但注意 `openai`（小写）是官方保留字，不能用于自定义 Provider。

> 来源：1962900#1 @yongye「Codex App 切到 API Provider 后历史会话消失解决办法」

方案 3：codex-provider-sync 工具

功能：一键同步 rollout 元数据和 SQLite 里的 provider，让 CLI / App 都能看到同一批历史会话。

安装 & 使用：

```
# 安装
npm install -g github:Dailin521/codex-provider-sync

# 查看当前状态
codex-provider status

# 同步 Provider
codex-provider sync

# 重启 Codex App
```

支持：Windows / macOS / Linux

> 来源：2293227#1 @wanghengtong「Codex APP切换 Provider 后，历史对话全没了，恢复办法」

GitHub：https://github.com/Dailin521/codex-provider-sync

方案 4：Codex-Threadripper 工具

功能：自动撕裂线程之间的隔阂，让所有 Provider 下的会话都能看到。支持后台服务持续运行。

安装 & 使用：

```
# 安装 (macOS/Linux)
brew tap wangnov/tap
brew install codex-threadripper

# 查看状态
codex-threadripper status

# 启动后台服务（自动同步）
codex-threadripper start
```

⚠ 注意：Codex 128 版本引入了新行为：恢复会话时自动切换到上一个会话使用的 Provider。工具可能需要更新以适配。

> “来源：2120343#1 @wangnov「【开源】切 provider 避免丢会话：Codex-Threadripper」

GitHub：https://github.com/Wangnov/codex-threadripper

方案 5：CLI 直接 resume 会话

适用场景：如果只是临时需要访问某个历史会话，可以直接用 CLI 的 resume 命令。

操作步骤：

1. 在 CC Switch 的会话管理中找到目标会话的 ID
2. 直接运行命令

```
codex resume 019de7eb-623c-7d12-938b-c34026f8d6fd
```

> “来源：2098635#1 @hwj1

四、最佳实践（预防措施）

1. 统一 Provider 名称

在配置所有中转站时，统一使用同一个 Provider 名称（如 `custom`），这样切换时不会丢失会话。

2. 使用 "OpenAI"（大写）作为 Provider 名称

```
model_provider = "OpenAI"
[model_providers.OpenAI]
name = "openai"
base_url = "http://127.0.0.1:8137/v1"
```

注意：这个方法对公益站有效，但官方账号登录后不会显示第三方会话。且 `openai`（小写）是保留字，不能用。

> 来源：2120343#17 @Lu_Hang

3. 定期备份

切换 Provider 前，备份 `~/.codex/` 目录，尤其是 `state_5.sqlite` 和 `sessions/` 文件夹。

五、相关工具汇总

工具名称	功能	GitHub	来源
codex-provider-sync	同步 rollout 元数据和 SQLite provider	GitHub	2293227
Codex-Threadripper	自动撕裂线程隔阂，支持后台服务	GitHub	2120343
CodexPlusPlus	Codex App 增强工具，含迁移功能	GitHub	2293227
CC Switch	跨平台桌面 All-in-One 助手	GitHub	2098635

六、关键帖子索引

帖子	标题	日期	方案类型
2293227	Codex APP切换 Provider后，历史对话全没了，恢复办法	2026-06-02	codex-provider-sync
2139733	Codex + CC Switch 切换供应商后 codex会话消失 /401 的完整修法	2026-05-09	完整修法
2120343	【开源】切 provider 避免丢会话：Codex-Threadripper	2026-05-06	Threadripper 工具

帖子	标题	日期	方案类型
2098635	分享一个10秒钟解决ccswitch切换中转站导致codex历史会话消失的问题	2026-05-02	统一Provider名称
1962900	Codex App切到API Provider后历史会话消失解决办法	2026-04-14	修改SQLite
1926891	Codex从中转切换到官方如何恢复session	2026-04-08	恢复方法
2265792	codex远程压缩对话历史切供应商消失如何解决	2026-06-01	完全解惑
2234221	Codex app官方/第三方provider切换指南	2026-05-24	切换指南
1717518	使用codex迁移自定义provider会话至openai team账号的提示词	2026-03-10	迁移提示词

如何与 Codex 协作？

1. 建立项目
创建独立的项目文件夹，放入已有代码、文档等材料；初始化 Git，并提交项目的初始版本。

推荐使用 VS Code 配合 Git。安装、初始化及基本操作可以直接询问 ChatGPT，无需一次掌握 Git 的全部高级功能。

2. 阅读并确认
先让 Codex 阅读项目中的相关材料，不要立即修改代码。

相关材料可以包括：

已有源代码和算例；
论文、理论资料和经典教材；
接口说明、输入文件及历史记录；
已有的操作手册或项目规范。

同时，与 Codex 确认关键约定，例如：

节点和自由度的编号顺序；
坐标系与正方向；
单位制、接口和数据格式；
哪些已有功能必须保持不变。

建议与 Codex 共同维护一份 Markdown 文档，持续记录讨论结论。 Markdown 既便于 Codex 修改，也便于人工阅读；也可以使用 .txt、.tex、.json 等纯文本文件。

3. 根据任务规划方案
根据具体任务，与 Codex 讨论代码的新增、修改或升级方案，并持续记录在文档中，反复迭代确认。

规划阶段就应明确代码的验证方法，例如：

1. 解析解对比： 让 Codex 根据可靠资料整理解析解，并与数值结果进行比较。
2. 退化或回归对比： 例如将三维程序退化到二维条件，与已验证的二维程序比较计算结果。
3. 数值差分核对： 使用有限差分对解析灵敏度进行独立核对。
4. 其他验证方式等。

4. 明确协作约束
正式执行前，应向 Codex 说明代码风格和修改边界，例如：

修改集中在指定文件；
只做完成任务所需的最小修改；
不随意重构无关模块；
不增加不必要的防御性代码；
不保留不需要的向后兼容；
不擅自增加依赖或改变接口；
发现约定不明确时先说明，不进行静默假设。

这些规则来自长期协作经验，可以保存在 AGENTS.md、个人操作手册或 Playbook 等文档中，供不同项目复用。

5. 修改并验证
让 Codex 按照已经确认的方案执行修改，并自行运行验证。

请严格按照已经确认的代码修改方案，在既定约束下完成修改，并在完成后执行全部验证 “
。

6. 使用 Git 管理与审阅代码
Codex 完成修改后，不要只阅读它的总结，也不要仅因 “测试通过 ”就直接接受，应使用 Git 查看实际代码差异。

重点审阅：

哪些文件被修改；
修改前后有哪些具体差异；
是否遵守既定协作约束；
是否出现无关重构；
是否增加了不必要的依赖或复杂度。

需要更详细的安装、仓库管理、提交与代码差异审阅步骤时，可参考 课题组 Git 使用资料（需登录） 与菜鸟教程： Git 教程 。建议先掌握查看修改、提交版本与恢复历史版本，再按实际需要学习分支和远程协作。

7. 决定是否接受更改
修改合理：提交 Git 版本；
存在局部问题：要求 Codex 根据差异继续修正；
修改方向错误：回滚到修改前的版本。

核心流程

创建项目并初始化 Git
→ 阅读代码与理论材料
→ 共同维护文档
→ 讨论并形成代码方案
→ 明确修改边界和编码习惯
→ Codex 修改并验证
→ 查看 Git 差异
→ 修正、提交或回滚