title: "Codex Windows 启动后只有后台进程、没有窗口：排查与修复指南"
author: "社区故障记录 / Workaround"
date: "2026-09-12"

适用场景

本文适用于 **Codex / ChatGPT Windows 桌面版**出现以下典型现象时：

- 点击启动后没有任何可见窗口；
- 任务管理器里能看到多个 `ChatGPT.exe`；
- 这些进程长期存活，甚至持续占用 CPU / 内存；
- `MainWindowHandle = 0`、`MainWindowTitle` 为空；
- 进程树里可以看到 GPU、NetworkService、StorageService、Crashpad，但 **没有 `--type=renderer`**；
- 重装、Repair / Reset、清理 Chromium profile 后仍然复现；
- `%LOCALAPPDATA%\OpenAI\Codex\runtimes\cua_node` 下不断出现 `.staging-*` 目录。

> **一句话概括：**安装包本身通常是完整的，但 Codex 第一次启动或版本更新后，需要把随 MSIX 安装包附带的 `cua_node` runtime 从 `WindowsApps` 搬到用户目录。某些 Windows 环境下，这个 relocation / staging 过程会在复制受保护文件时失败，留下半成品 runtime，最终导致 app-server / renderer 没有启动，桌面窗口也就不会出现。

已验证案例

本文方法已在以下环境中实际验证：

- Codex Windows Stable：`26.908.4834.0`
- 平台：Windows 11 x64
- MSIX package：`OpenAI.Codex`

该版本安装包中的官方 `cua_node` runtime：

```
SourceFiles : 4123
Manifest    : True
NodeExe     : True
NodeRepl    : True
```

但自动 staging 后出现：

```
StagingFiles : 517
MissingFiles : 3606
NodeExe      : True
NodeRepl     : False
```

手工使用 `xcopy /G` 将当前版本完整 runtime 放到正确的 content-addressed runtime 目录后：

```
Copied 4123 files
XCOPY exit code: 0

SourceFiles : 4123
TargetFiles : 4123
Manifest    : True
NodeExe     : True
NodeRepl    : True
Missing files: 0
Extra files:   0
```

随后 Codex **立即正常启动并显示 GUI**。

为什么会发生

Microsoft Store / MSIX 包中的部分文件可能带有 Windows 的 **Application Protected** 属性。

典型例子是：

```
bin\node_repl.exe
```

这类文件从：

```
C:\Program Files\WindowsApps\...
```

复制到普通用户目录时，并不总能使用普通文件复制语义。已有公开报告显示，普通 `Copy-Item` / Node `copyfile` 可能返回：

```
The specified file could not be encrypted.
HRESULT: 0x80071770
```

而 Windows 的：

```
xcopy /G
```

允许“加密 / Application Protected 源文件复制到解密后的普通目标位置”，因此可以完成这一步。

Codex 当前某些 Windows 版本的 runtime materialization 流程没有可靠处理这种失败，于是会形成：

```
完整 MSIX runtime
      |
      v
创建 .staging-<runtime-id>-<random>
      |
      v
复制一部分文件
      |
      v
遇到 Application Protected 文件 -> 失败
      |
      v
staging 未完成 / 未 finalize
      |
      v
app-server / renderer 未启动
      |
      v
ChatGPT.exe 挂在后台，但没有窗口
```

因此：重装往往无效。重装只是重新获得一份完整 MSIX 包；下一次启动仍然会再次走同一套有问题的 relocation 流程。

第一步：确认是否属于同一类故障

先正常启动 Codex，等十几秒，然后打开 PowerShell 7。

1. 检查窗口句柄

```
Get-Process ChatGPT |
Select Id, CPU, WorkingSet, Responding, MainWindowHandle,
MainWindowTitle
```

如果多条进程都是：

```
Responding       : True
MainWindowHandle : 0
MainWindowTitle  :
```

说明进程确实起来了，但没有创建正常顶层窗口。

2. 检查进程类型

```
$pkg = Get-AppxPackage OpenAI.Codex

Get-CimInstance Win32_Process |
Where-Object {
    $_.ExecutablePath -like "$($pkg.InstallLocation)*"
} |
Select ProcessId, Name, CommandLine |
Format-List
```

故障状态常见：

```
ChatGPT.exe
ChatGPT.exe --type=crashpad-handler
ChatGPT.exe --type=gpu-process
ChatGPT.exe --type=utility --utility-sub-
type=network.mojom.NetworkService
ChatGPT.exe --type=utility --utility-sub-
type=storage.mojom.StorageService
```

但看不到：

```
ChatGPT.exe --type=renderer
```

这时继续检查 runtime。

第二步：检查 `cua_node` staging 是否残缺

运行：

```
$runtimeRoot = "$env:LOCALAPPDATA\OpenAI\Codex\runtimes\cua_node"

Get-ChildItem $runtimeRoot -Directory -Force -ErrorAction
SilentlyContinue |
Sort-Object LastWriteTime -Descending |
ForEach-Object {
    [PSCustomObject]@{
        Name     = $_.Name
        Files    = (Get-ChildItem $_.FullName -Recurse -File -Force -
ErrorAction SilentlyContinue).Count
        NodeExe  = Test-Path "$($_.FullName)\bin\node.exe"
        NodeRepl = Test-Path "$($_.FullName)\bin\node_repl.exe"
    }
}
```

如果看到大量类似：

```
.staging-a708e72b10c27b59-XXXXXX   517    True    False
.staging-a708e72b10c27b59-XXXXXX  3450    True    False
.staging-xxxxxxxxxxxxxxxx-XXXXXX    93    True    False
```

尤其是：

```
NodeExe  = True
NodeRepl = False
```

就与本文故障高度吻合。

> 注意：不同 Codex 版本的 runtime ID 和完整文件数会变化。不要照抄本文的 `a708e72b10c27b59` 或 `4123`，应以你当前版本实际检测结果为准。

第三步：确认安装包里的官方 runtime 是完整的

```
$pkg = Get-AppxPackage OpenAI.Codex
$srcRoot = Join-Path $pkg.InstallLocation "app\resources\cua_node"

[PSCustomObject]@{
    PackageVersion = $pkg.Version
    SourcePath     = $srcRoot
    SourceExists   = Test-Path $srcRoot
    Files          = (Get-ChildItem $srcRoot -Recurse -File -Force -
ErrorAction SilentlyContinue).Count
    NodeExe        = Test-Path "$srcRoot\bin\node.exe"
    NodeRepl       = Test-Path "$srcRoot\bin\node_repl.exe"
}
```

理想状态：

```
SourceExists : True
NodeExe      : True
NodeRepl     : True
```

如果安装包里的 `node_repl.exe` 本身都不存在，就不要继续使用本文方法。

第四步：自动识别当前 runtime ID

使用最新的 `.staging-*` 目录来识别当前版本期待的 runtime ID：

```
$runtimeRoot = "$env:LOCALAPPDATA\OpenAI\Codex\runtimes\cua_node"

$latestStaging = Get-ChildItem $runtimeRoot -Directory -Force |
Where-Object { $_.Name -like ".staging-*" } |
Sort-Object LastWriteTime -Descending |
Select-Object -First 1

if (-not $latestStaging) {
    throw "没有找到 .staging-* 目录，本文方法可能不适用于当前故障。"
}

if ($latestStaging.Name -notmatch '^\.staging-([0-9a-fA-F]+)-') {
    throw "无法从 staging 目录名解析 runtime ID：$($latestStaging.Name)"
}

$runtimeId = $Matches[1]
$runtimeId
```

输出示例：

```
a708e72b10c27b59
```

第五步：执行修复

> **安全说明：**下面操作不会修改 `WindowsApps` 中的官方安装包，只会把官方包内现成的 runtime 复制到当前用户的 runtime cache。建议保留 `.staging-*`，等确认修复成功后再清理。

1. 彻底停止当前 Codex / ChatGPT 进程

```
$pkg = Get-AppxPackage OpenAI.Codex

Get-CimInstance Win32_Process |
Where-Object {
    $_.ExecutablePath -like "$($pkg.InstallLocation)*"
} |
ForEach-Object {
    Stop-Process -Id $_.ProcessId -Force -ErrorAction SilentlyContinue
}

Start-Sleep -Seconds 3
```

确认没有残留：

```
Get-CimInstance Win32_Process |
Where-Object {
    $_.ExecutablePath -like "$($pkg.InstallLocation)*"
}
```

应当没有输出。

2. 设置源目录和目标目录

```
$srcRoot = Join-Path $pkg.InstallLocation "app\resources\cua_node"
$runtimeRoot = "$env:LOCALAPPDATA\OpenAI\Codex\runtimes\cua_node"
$target = Join-Path $runtimeRoot $runtimeId
```

如果 `$target` 已存在但怀疑不完整，建议先备份：

```
if (Test-Path $target) {
    $backup = "$target.backup-$(Get-Date -Format 'yyyyMMdd-HHmmss')"
    Rename-Item $target $backup
    Write-Host "Existing target backed up to: $backup"
}
```

创建目标目录：

```
New-Item -ItemType Directory -Path $target -Force | Out-Null
```

3. 使用 `xcopy /G` 复制完整 runtime

```
xcopy "$srcRoot\*" "$target\" /E /H /K /G /Y /I
```

参数说明：

```
/E   复制所有子目录，包括空目录
/H   包含隐藏文件和系统文件
/K   保留文件属性
/G   允许加密 / 受保护源文件复制到解密目标
/Y   覆盖时不再询问
/I   将目标视为目录
```

如果最后显示：

```
XXXX File(s) copied
```

并且：

```
XCOPY exit code: 0
```

即可进入验证阶段。

第六步：完整验证

不要仅检查 `node_repl.exe` 是否存在。建议同时验证：

1. 源 / 目标文件数量；
2. 相对路径集合是否一致；
3. 关键文件 SHA-256 是否一致。

运行：

```
$srcFiles = Get-ChildItem $srcRoot -Recurse -File -Force
$dstFiles = Get-ChildItem $target  -Recurse -File -Force

$srcRelative = $srcFiles | ForEach-Object {
    $_.FullName.Substring($srcRoot.Length).TrimStart('\')
}

$dstRelative = $dstFiles | ForEach-Object {
    $_.FullName.Substring($target.Length).TrimStart('\')
}

$diff = Compare-Object $srcRelative $dstRelative

$missing = $diff |
Where-Object { $_.SideIndicator -eq '<=' } |
Select-Object -ExpandProperty InputObject

$extra = $diff |
Where-Object { $_.SideIndicator -eq '=>' } |
Select-Object -ExpandProperty InputObject

[PSCustomObject]@{
    SourceFiles = $srcFiles.Count
    TargetFiles = $dstFiles.Count
    Manifest    = Test-Path "$target\manifest.json"
    NodeExe     = Test-Path "$target\bin\node.exe"
    NodeRepl    = Test-Path "$target\bin\node_repl.exe"
    MissingFiles = $missing.Count
    ExtraFiles   = $extra.Count
}
```

关键文件 Hash：

```
$keyFiles = @(
    "manifest.json",
    "bin\node.exe",
    "bin\node_repl.exe"
)

foreach ($file in $keyFiles) {
    $src = Join-Path $srcRoot $file
    $dst = Join-Path $target $file

    [PSCustomObject]@{
        File  = $file
        Match = ((Get-FileHash $src -Algorithm SHA256).Hash -eq
                 (Get-FileHash $dst -Algorithm SHA256).Hash)
    }
}
```

理想结果：

```
SourceFiles  = TargetFiles
Manifest     = True
NodeExe      = True
NodeRepl     = True
MissingFiles = 0
ExtraFiles   = 0
```

并且关键文件全部：

```
Match = True
```

第七步：启动验证

正常从开始菜单启动 Codex。

如果修复命中，通常会出现明显变化：

- GUI 正常显示；
- `ChatGPT.exe --type=renderer` 出现；
- Codex app-server 正常启动；
- `MainWindowHandle` 不再全部为 0。

建议成功后：

1. 正常关闭 Codex；
2. 再次启动一次；
3. 确认第二次仍能正常打开。

一键修复脚本（适合已经确认故障指纹的用户）

> 运行前请先确认本文前面的故障指纹与你一致。脚本会自动读取当前 MSIX、最新 staging 的 runtime ID，并复制当前官方 runtime。

```
$ErrorActionPreference = "Stop"

$pkg = Get-AppxPackage OpenAI.Codex
if (-not $pkg) {
    throw "未找到 OpenAI.Codex MSIX 包。"
}

$srcRoot = Join-Path $pkg.InstallLocation "app\resources\cua_node"
$runtimeRoot = "$env:LOCALAPPDATA\OpenAI\Codex\runtimes\cua_node"

if (-not (Test-Path $srcRoot)) {
    throw "官方 cua_node runtime 不存在：$srcRoot"
}

$latestStaging = Get-ChildItem $runtimeRoot -Directory -Force |
Where-Object { $_.Name -like ".staging-*" } |
Sort-Object LastWriteTime -Descending |
Select-Object -First 1

if (-not $latestStaging) {
    throw "没有找到 .staging-*，无法自动识别 runtime ID。"
}

if ($latestStaging.Name -notmatch '^\.staging-([0-9a-fA-F]+)-') {
    throw "无法解析 runtime ID：$($latestStaging.Name)"
}

$runtimeId = $Matches[1]
$target = Join-Path $runtimeRoot $runtimeId

Write-Host "Package version: $($pkg.Version)"
Write-Host "Runtime ID:      $runtimeId"
Write-Host "Source:          $srcRoot"
Write-Host "Target:          $target"

# 停止当前包中的所有进程
Get-CimInstance Win32_Process |
```

```
Where-Object { $_.ExecutablePath -like "$($pkg.InstallLocation)*" } |
ForEach-Object {
    Stop-Process -Id $_.ProcessId -Force -ErrorAction SilentlyContinue
}

Start-Sleep -Seconds 3

$left = Get-CimInstance Win32_Process |
Where-Object { $_.ExecutablePath -like "$($pkg.InstallLocation)*" }

if ($left) {
    throw "仍有 Codex / ChatGPT 进程未退出。"
}

# 若目标已存在则备份
if (Test-Path $target) {
    $backup = "$target.backup-$(Get-Date -Format 'yyyyMMdd-HHmmss')"
    Rename-Item $target $backup
    Write-Host "Old target backed up: $backup"
}

New-Item -ItemType Directory -Path $target -Force | Out-Null

# 复制官方 runtime；/G 是关键
xcopy "$srcRoot\*" "$target\" /E /H /K /G /Y /I
$xcopyCode = $LASTEXITCODE

if ($xcopyCode -gt 1) {
    throw "XCOPY failed, exit code: $xcopyCode"
}

# 验证
$srcFiles = Get-ChildItem $srcRoot -Recurse -File -Force
$dstFiles = Get-ChildItem $target  -Recurse -File -Force

$srcRelative = $srcFiles | ForEach-Object {
    $_.FullName.Substring($srcRoot.Length).TrimStart('\')
```

```
}

$dstRelative = $dstFiles | ForEach-Object {
    $_.FullName.Substring($target.Length).TrimStart('\')
}

$diff = Compare-Object $srcRelative $dstRelative
$missing = @($diff | Where-Object { $_.SideIndicator -eq '<=' })
$extra   = @($diff | Where-Object { $_.SideIndicator -eq '=>' })

$ok = (
    $srcFiles.Count -eq $dstFiles.Count -and
    $missing.Count -eq 0 -and
    $extra.Count -eq 0 -and
    (Test-Path "$target\manifest.json") -and
    (Test-Path "$target\bin\node.exe") -and
    (Test-Path "$target\bin\node_repl.exe")
)

Write-Host ""
Write-Host "SourceFiles : $($srcFiles.Count)"
Write-Host "TargetFiles : $($dstFiles.Count)"
Write-Host "Missing     : $($missing.Count)"
Write-Host "Extra       : $($extra.Count)"
Write-Host "Manifest    : $(Test-Path "$target\manifest.json")"
Write-Host "NodeExe     : $(Test-Path "$target\bin\node.exe")"
Write-Host "NodeRepl    : $(Test-Path "$target\bin\node_repl.exe")"

if ($ok) {
    Write-Host ""
    Write-Host "RUNTIME VERIFY PASSED. 现在可以正常启动 Codex。"
} else {
    throw "Runtime verification failed. 暂时不要启动 Codex。"
}
```

更新后会不会再次出现

有可能。

Codex 的 `cua_node` 内容更新后，content-addressed runtime ID 也可能改变。

例如：

```
旧版本 runtime ID -> 440c4f095d41ea30
新版本 runtime ID -> a708e72b10c27b59
```

旧 runtime 即使完整，也不能替代新版本所需 runtime。

因此，如果官方尚未修复 relocation 逻辑，那么每次更新到新的 runtime ID 后，都可能再次出现：

```
后台进程存在
+ 无 GUI
+ 无 renderer
+ 新 .staging-* 不断出现
+ NodeExe=True / NodeRepl=False
```

此时重新执行本文方法即可。

成功后是否可以删除 `.staging-*`

建议先确认：

1. Codex 能正常启动；
2. 关闭后再次启动仍正常；
3. 当前正式 runtime 目录完整存在。

然后可以考虑清理旧的 `.staging-*` 半成品目录以回收空间。

不要删除当前已经正常工作的正式 runtime ID 目录。

如果某些 staging 目录提示文件正在被占用，请先完全退出 Codex / ChatGPT 再操作。

不建议优先折腾的方向

如果你的故障指纹与本文一致，以下操作通常不是根因修复：

- 反复卸载 / 重装；
- 重置 Chromium profile；
- 清登录状态；
- 重装 WebView2；
- 反复更新显卡驱动；
- 强行 `ShowWindow()`；
- 只补一个旧版本的 runtime；
- 从网上下载第三方 runtime 覆盖。

本文 workaround 使用的是 **当前已安装官方 MSIX 包中自带的 runtime**，不需要下载任何第三方二进制。

公开 Issue 参考

本文现象与以下公开报告高度相关：

- 当前验证案例：`openai/codex#45000`
  https://github.com/openai/codex/issues/45000
- `openai/codex#41540` - Application Protected `node_repl.exe` relocation failure / `0x80071770`
  https://github.com/openai/codex/issues/41540
- `openai/codex#41654` - runtime relocation / staging failure，完整 runtime 手工恢复后 GUI 正常
  https://github.com/openai/codex/issues/41654
- `openai/codex#42501` - 26.901.x 中同类 `cua_node` staging 问题，`xcopy /G` workaround
  https://github.com/openai/codex/issues/42501
- `openai/codex#40843` - WindowsApps / Application Protected 文件 relocation 与 `ERROR_ENCRYPTION_FAILED`
  https://github.com/openai/codex/issues/40843

最后总结

如果只记一件事：

> **Codex Windows 版“后台进程正常，但没有窗口”不一定是 Electron / GPU / profile 问题。**
> 如果同时存在大量残缺 `.staging-*`，尤其 `node.exe` 已复制而 `node_repl.exe` 缺失，那么很可能是 MSIX 中 Application Protected runtime 向用户目录 relocation 失败。

核心修复逻辑：

```
识别当前 runtime ID
      ->
停止 Codex
      ->
从当前官方 MSIX 包读取完整 app\resources\cua_node
      ->
使用 xcopy /G 复制到正确的 runtime ID 目录
      ->
验证文件数 / 路径 / Hash
      ->
重新启动 Codex
```

在已验证案例中，完成这一过程后 Codex **秒启恢复正常**。