

WLSAPlus macOS 源码构建包说明书（1.0.9）

包里有什么

下载 `wlsaplust1.0.9.zip`，解压后得到 `wlsaplust1.0.9` 文件夹，里面有：

文件	说明
<code>working.command</code>	一键构建脚本：自动安装依赖 → 构建 → 签名 → 解除隔离
<code>wlsaplustformacos</code>	WLSAPlus 源码（GitHub 仓库的完整克隆，已包含 macOS VPN 内核 mihomo）
<code>note.pdf</code> / <code>note.md</code>	本说明书

请保持这个结构：`working.command` 和 `wlsaplustformacos` 文件夹要放在同一个目录里。

建议把整个 `wlsaplust1.0.9` 文件夹移到**个人文件夹**（访达 → 前往 → 个人，即 `~/wlsaplust1.0.9`）再构建。放在「下载」「桌面」或「文稿」里也可以，但 macOS 会弹出「“终端”想访问“下载”文件夹中的文件」之类的提示，请点**「允许」**，否则脚本无法读写文件。

用哪个脚本？ 只用本包里的 `working.command`。源码里的 `release-kit/legacy-构建wlsaplust.command` 是已弃用的旧脚本（`npm install + electron:make`，依赖 Forge maker-dmg / appdmg 原生模块），请不要双击它。`working.command` 用 `electron-forge package` 打包，再用 macOS 自带的 `hdiutil` 生成 DMG（不用 Forge 的 maker-dmg）。

系统要求

- Mac 使用 **Apple 芯片**（M1/M2/M3/M4…）。Intel Mac 不支持。
- **macOS 13（Ventura）或更新版本**（Electron 44 要求；更老的系统脚本会直接提示版本过低）。
- 全新的 Mac 也可以：**不需要**事先装 Node、Xcode，脚本会自动装好（Homebrew 一般用不到，只在备用时才装）。
- 需要联网，但**不需要科学上网**：所有下载都走国内镜像（见下表）。
- 硬盘至少留出约 **5 GB** 空间（脚本开始时会检查）；首次构建约 **10-30 分钟**（取决于网速）。

软件	下载来源
Xcode 命令行工具	Apple 官方（系统弹窗安装，国内可直连）
nvm	Gitee（ gitee.com/mirrors/nvm ，固定版本 v0.40.8）
Node.js 22	npmmirror
npm 依赖、Electron	npmmirror
Homebrew（仅备用）	只有 nvm 装 Node.js 失败时才安装：清华 TUNA 镜像（失败自动换中科大 USTC）
VPN 内核（mihomo）	已在源码 electron/bin/mac-vpn.tar.gz 中，无需下载

构建步骤

1. 双击 `wlsaplus1.0.9.zip` 解压。
2. 打开 `wlsaplus1.0.9` 文件夹，双击 `working.command`。
 - 若提示「无法打开，因为无法验证开发者」：点「完成」→ 打开「系统设置」→「隐私与安全性」→ 拉到最下面点「仍要打开」→ 再点「打开」。
 - 若提示「没有执行权限」：打开「终端」，输入 `chmod +x`（后面有个空格），把 `working.command` 拖进终端窗口，按回车，再双击脚本。
3. 脚本会自动完成以下步骤（窗口里会显示 0/8 … 8/8 进度）：
 1. 检查系统和芯片；
 2. 检查 **Xcode 命令行工具**，没有就弹窗安装 —— 请点「安装」并同意协议，装好后脚本自动继续；
 3. 从 Gitee 安装 **nvm**，再从 npmmirror 安装 **Node.js 22**（只有这一步失败时，才会从国内镜像安装 **Homebrew** 来装 Node.js）；
 4. 把 npm 和 Electron 的下载源设为 **npmmirror**；
 5. 检查源码和 VPN 内核（mihomo）是否完整；
 6. 安装 npm 依赖；
 7. 构建界面；
 8. 打包 **arm64** 应用，做 **ad-hoc 签名**，并解除 macOS 隔离属性；
 9. 在 `output` 文件夹生成 **WLSAPlus.app** 和 **wlsaplus1.0.9.dmg**（用 `hdiutil` 生成），自动在访达中显示，并询问是否立即打开。若 DMG 生成失败，脚本会明确提示，此时直接把 `output/WLSAPlus.app` 拖进「应用程序」即可。
4. 正常情况下全程**不需要输入密码**。只有用到备用的 Homebrew 时才会要求输入开机密码：输入时屏幕**不会显示任何字符**，输完直接按回车即可；若提示 `Press RETURN to continue`，按一下回车。

构建好之后安装

1. 打开 `output/wlsaplus1.0.9.dmg`，把 **WLSAPlus** 拖进「应用程序」。（也可以直接双击 `output/WLSAPlus.app` 使用。）
2. 第一次打开时，在「应用程序」里双击 WLSAPlus。若提示「无法验证开发者」/「Apple 无法检查其是否包含恶意软件」：点「完成」→ 打开「系统设置」→「隐私与安全性」，拉到最下面，在 WLSAPlus 那一行点「仍要打开」，输入密码后再点「打开」。（macOS 15 起，「右键 → 打开」已经不能绕过这个拦截，请按上面的方法操作。）

如果提示「已损坏」或打不开

本应用是自签名的（ad-hoc，不是苹果开发者证书，也没有经过公证），个别系统仍会拦。

1. 先试「系统设置」→「隐私与安全性」→「仍要打开」（见上一节）；
2. 若提示「已损坏，无法打开」，在「终端」执行下面这行（不需要 `sudo`），然后再打开：

```
xattr -cr /Applications/WLSAPlus.app
```

构建失败怎么办

- **网络波动**（npm、Node 下载失败）：直接重新双击 `working.command`，已经装好的部分会自动跳过。
- **一直卡在“等待命令行工具安装完成”**：确认已在弹窗里点了「安装」；也可以在终端执行 `xcode-select --install`。
- **提示“尚未同意 Xcode 许可协议”**（装过完整 Xcode 的 Mac 常见）：在终端执行 `sudo xcodebuild -license accept`，输入开机密码后按回车，再重新双击脚本。
- **提示 macOS 版本过低**：WLSAPlus 需要 macOS 13 或更新版本，请先升级系统。
- **提示在 Rosetta 模式下运行**：在「应用程序 → 实用工具」里右键「终端」→ 显示简介，取消「使用 Rosetta 打开」。
- **提示 VPN 内核缺失或已损坏**（`electron/bin/mac-vpn.tar.gz`）：重新下载并解压完整的 `wlsaplus1.0.9.zip`，不要单独挪动 `wlsaplusformacos` 里的文件。
- 其它问题：把 `output/build.log`（完整构建日志）发给我们。

使用 WLSAPlus 论坛（1.0.9 新增）

左侧栏（窗口较窄时为底部栏）有 **Forum / WLSAPlus 论坛** 标签，论坛直接嵌在应用里打开：

- **自动登录**：已在应用里同步 PowerSchool 的同学，打开论坛会用当前的 PowerSchool 会话自动登录（密码不会发给论坛）；帖子依旧完全匿名。未同步时以游客身份浏览。
- **主题跟随**：论坛的浅色 / 深色跟随 WLSAPlus 的主题设置（设置里选「跟随系统」时随系统切换），切换后论坛即时变化，无需刷新。

- **底部导航**：论坛的「主页 / AI 助手 / 墙投稿 / 墙查询 / 发布 / 登录」在论坛页面底部；窗口拉宽时内容居中显示。
- 自动登录偶尔失败时，页面底部会短暂提示错误代码（如 `invalid_session`），可截图反馈；不影响以游客身份浏览。
- 退出或切换 PowerSchool 账号时，论坛登录状态会一并清除。

使用 VPN

1. 打开 Tools → VPN。
2. 可先「Refresh」节点列表，需要时点「Test latency」测延迟，再选中节点。
3. 点 Connect。若弹出管理员授权，输入密码并允许。授权前若短暂提示等待，属正常，不要反复乱点。
4. 可用「Test WeChat」检测连 VPN / 不连 VPN 时微信相关地址是否可达（仅网络探测，不等于微信能否登录）。

关闭 VPN（请按顺序）

1. 回到 VPN 页，若有 Disconnect，先点 Disconnect。
2. 打开「活动监视器」→「网络」，找到 `clash`（或 `mihomo` 相关进程），选中后点左上角停止（叉），确认退出。
3. 确认 VPN 页状态回到 Ready / Connect。

开课提醒

设置里可开关「Class reminders」。开启后，开课前约 5 分钟会发 macOS 通知（需已同步课表）。

课表与成绩

Connect / 同步 PowerSchool 后，可在课表与 Progress（成绩、作业、考勤）中查看。数据保存在本地。

附：脚本对系统做了什么

- 安装 Xcode 命令行工具、`nvm`（`~/nvm`）和 Node.js 22；
- 若 `~/.zshrc` 里还没有 `nvm` 设置，在末尾写入 `nvm` 环境和 `npmmirror` 镜像（以 `WLSAPlus nvm` 标记，可手动删除）；
- 把 `npm` 下载源设为 `https://registry.npmmirror.com`（写入 `~/.npmrc`）；
- 只有 `nvm` 失败、需要备用 Homebrew 时：安装 Homebrew（`/opt/homebrew`），并在 `~/.zprofile` 末尾写入 `brew` 环境和国内镜像设置（以 `WLSAPlus Homebrew mirror` 标记，可手动删除）；

- 构建产物只放在本文件夹的 `wlsaplusformacos/out` 和 `output` 里，不会改动「应用程序」，需要你自己把 WLSAPlus 拖进去。

附：给开发者的构建说明

- 签名：Electron Forge 不做签名（`forge.config.cjs` 中 `ad-hoc` 构建不设 `osxSign`），由 `working.command` 在打包后统一执行 `codesign --force --deep --sign - --entitlements electron/entitlements.plist` 做 `ad-hoc` 签名。
- DMG：由 macOS 自带的 `hdiutil` 生成，不使用 Forge 的 `maker-dmg` (`appdmg`)。
- `package.json` 里的 `overrides` 把 `@electron/node-gyp` 固定为 npm 上发布的 `10.2.0-electron.1`：否则 `@electron/rebuild` 会从 GitHub 拉取 `node-gyp`（国内经常失败）。请不要删除这条 `override`。
- 国内镜像相关：脚本设置 `npm_config_replace_registry_host=npmjs`，让锁文件里的 `registry.npmjs.org` 地址一律改走 `npmmirror`；`npm ci` 编译原生模块时用本机 Node 头文件（缺失时从 `npmmirror` 下载），打包前会清掉这些设置，避免影响 `@electron/rebuild`。