**面向开发工具的AI插件中英本地化指南与质量检查清单**

**超越翻译，实现原生体验**

本地化的目标不是简单地将中文界面翻译成英文，而是要创造一款让英语世界的开发者感觉“就是为我而生”的产品。这意味着每一个词汇、每一条提示、甚至每一个标点符号，都必须符合目标用户的语言习惯、技术语境和文化预期。本地化是产品开发不可或割的一部分，而非事后补救。

本指南分为三个核心部分：

1. **第一阶段：战略准备与基础建设 (面向项目经理与负责人)**
2. **第二阶段：本地化执行核心检查清单 (面向技术翻译、工程师与测试人员)**
3. **第三阶段：质量保证与持续迭代**

**第一阶段：战略准备与基础建设**

在开始翻译工作之前，必须完成以下准备工作。

* **1. 组建跨职能团队：**
  + **本地化项目经理 (Localization PM):** 负责流程、资源和时间线。
  + **母语为英语的技术翻译:** 不仅要懂语言，更要懂代码，最好本身就是开发者。
  + **本地化工程师:** 处理代码中的字符串提取、构建流程和技术问题。
  + **英语市场的开发者/产品顾问:** 作为最终用户代表，提供关于术语、功能和体验的反馈。
* **2. 创建术语库 (Termbase/Glossary):**
  + 这是保证一致性的基石。必须在项目启动时就建立，并持续更新。
  + **内容应包括：**
    - **产品特有功能词汇:** 如“智能代码生成”、“代码片段联想”、“一键重构”等。
    - **通用技术术语:** 如“仓库(Repository)”、“拉取请求(Pull Request)”、“依赖(Dependency)”。明确哪些翻译，哪些保留原文（如 API, JSON, Git）。
    - **UI界面元素:** 如“设置(Settings)”、“首选项(Preferences)”、“命令面板(Command Palette)”。
  + **示例:**  
    | 中文源词 | 推荐英文 | 状态 (Approved/Pending) | 注释 |  
    | :--- | :--- | :--- | :--- |  
    | 智能补全 | Autocomplete | Approved | 比 "Intelligent Completion" 更流行，更符合市场认知。 |  
    | 一键生成 | Generate | Approved | 在按钮等场景下，力求简洁。避免使用 "One-click Generate" 这种中式英语。 |  
    | 工作区 | Workspace | Approved | 遵循IDE的通用叫法。 |  
    | 提交 | Commit | Do Not Translate | 这是 Git 的核心命令，必须保留原文。 |
* **3. 制定本地化风格指南:**
  + 这是保证产品“说话方式”一致的规范。
  + **核心内容应定义：**
    - **语调:专业的、清晰**但不过于严肃或口语化。避免使用过于夸张的营销词汇。
    - **大小写规则:** 明确标题、按钮、菜单、标签等使用哪种大小写（见下文详述）。
    - **标点符号:** 统一使用英文标点。例如，使用英文句号 . 而非 。，使用半角逗号 ,。
    - **日期/时间格式:** 定义统一格式，如 MM/DD/YYYY 或 YYYY-MM-DD。
    - **动词使用:** 在UI操作中，多使用祈使句动词（如 Generate Code, Save Settings）。

**第二阶段：本地化执行核心检查清单 (The Core Checklist)**

这是技术翻译和测试人员日常工作的核心依据。

**A. UI/UX 界面文本**

| **检查类别** | **检查点** | **Bad** | **Good** |
| --- | --- | --- | --- |
| **1. 命令面板 (Command Palette)** | **格式:** 必须遵循 对象: 动作 或 功能组: 动作 的格式。<br>**大小写:** 使用标题大写 (Title Case)。 | Generate the code | AI: Generate Code 或 Code: Generate (遵循IDE现有风格) |
|  | **简洁性:** 命令应简短、明确，易于搜索。 | 使用AI能力为当前函数生成单元测试 | AI: Generate Unit Test for Function |
| **2. 菜单项 & 按钮** | **大小写:** 菜单项通常用标题大写 (Title Case)，按钮标签通常用标题大写或句首大写 (Sentence case)，需与目标IDE（如VS Code, JetBrains）的风格保持一致。 | 现在应用 (button) | Apply (简洁，动词明确) |
|  | **动词驱动:** 按钮标签应以动词开头，清晰指示操作。 | 确认更改 | Confirm Changes 或 Save |
| **3. 设置 (Settings) & 首选项 (Preferences)** | **大小写:** 描述性文本和标签使用句首大写 (Sentence case)。 | Enable The Automatic Code Completion | Enable automatic code completion (更自然) |
|  | **描述清晰:** 设置项的描述必须解释清楚该选项的作用，以及可能带来的影响。 | 开启智能模式 | Automatically suggest code snippets based on the current context. |
| **4. 提示与信息 (Tooltips, Info-boxes)** | **简洁有益:** Tooltip应提供光标悬停元素的额外、有价值的信息，而不是重复标签。 | 这是保存按钮 | Save the current file (Ctrl+S) (提供了快捷键信息) |
|  | **大小写:** 使用句首大写 (Sentence case)。 | Click Here To Learn More. | Click here to learn more. |
| **5. 占位符 (Placeholder Text)** | **引导性:** 输入框内的占位符应引导用户输入正确格式或类型的内容。 | 输入 | Enter your API key... |
| **6. 错误/成功/警告信息** | **人性化:** 避免使用生硬、机械的术语如“失败”、“错误代码: 500”。 | 执行失败 | Failed to generate code. Please check your network connection and try again. (解释原因，提供解决方案) |
|  | **语气恰当:** 成功信息应积极但不夸张。 | 恭喜！代码已生成！ | Code generated successfully. (专业且直接) |
| **7. 空状态 (Empty States)** | **引导操作:** 当列表或区域为空时，应引导用户进行第一步操作。 | 无数据 | No code snippets found. Start by generating your first one! (友好且具引导性) |

**B. 技术术语与内容**

| **检查类别** | **检查点** | **注意事项** |
| --- | --- | --- |
| **1. 术语一致性** | 检查所有界面、文档、日志中的同一术语是否使用了术语库中定义的唯一翻译。 | 使用CAT工具的QA功能或手动抽查。仓库 不能一会是 repository，一会是 warehouse。 |
| **2. 不可翻译内容 (DNT - Do Not Translate)** | 严格遵守不翻译原则，保护代码和命令的完整性。 | **绝对不能翻译的:** API, URL, JSON, XML, SDK, Git, Docker, commit, push, pull, true, false, null 等编程语言关键字和专有命令。 |
| **3. 代码示例与注释** | 代码示例中的变量名、函数名、字符串内容应本地化，使其对英语用户有意义。 | let 用户名 = "张三"; 应改为 let userName = "John Doe";。代码注释也必须翻译。 |
| **4. 文档与帮助链接** | 检查所有指向外部文档、教程或社区的链接。 | 必须将指向中文社区（如思否、掘金）的链接替换为全球开发者社区（如 Stack Overflow, GitHub Discussions, a relevant blog post on Medium）。 |
| **5. 正则表达式与快捷键** | 检查硬编码在代码中的正则表达式是否能正确处理英文字符。 | /[\\u4e00-\\u9fa5]/ 这种匹配中文的正则必须修改。快捷键 Ctrl+Alt+中 这种组合必须替换为不与常用软件冲突的英文章盘快捷键。 |

**C. 文化与格式**

| **检查类别** | **检查点** | **注意事项** |
| --- | --- | --- |
| **1. 文本扩展** | 英文翻译通常比中文长30%以上。必须检查所有UI元素，确保文本不会被截断或导致布局错乱。 | 这是最重要的UI测试点。在真实环境中（IDE内）进行全面的视觉检查。必要时需前端工程师调整UI。 |
| **2. 标点符号** | 确保所有标点均为英文半角符号。 | 检查 ， 。 ： （ ） 是否已全部替换为 , . : ( )。 |
| **3. 语气与文化** | 避免中式思维的直译。 | 请耐心等待... (Please wait patiently...) 可简化为 Loading... 或 Generating...。英文开发者社区偏爱直接和高效。 |
| **4. 图像与图标** | 检查UI中的图像或图标是否含有中文字符或不符合西方文化习惯的符号。 | 例如，用 ✓ (check mark) 表示成功，而不是一个笑脸。 |

**第三阶段：质量保证与持续迭代 (LQA - Linguistic Quality Assurance)**

* **1. 上下文内审校 (In-context Review):**
  + **禁止在Excel或字符串列表里做最终审核！** 译文必须被部署到测试环境中，在真实的IDE插件界面里进行审校。只有这样才能发现文本扩展、语境不符等问题。
* **2. 功能性测试:**
  + 本地化版本的功能必须与源语言版本完全一致。测试人员需要模拟真实用户场景，确保所有功能（特别是核心的AI功能）在英文环境下正常工作。
* **3. 建立反馈闭环:**
  + 为英语用户提供便捷的反馈渠道（如GitHub Issues, 邮件, 或内置反馈表单）。
  + 定期收集用户对术语、表述的反馈，并将其更新到术语库和风格指南中。本地化是一个持续优化的过程。
* **4. 自动化检查:**
  + 在CI/CD流程中集成自动化LQA检查工具，用于扫描：
    - **占位符错误:** 检查代码中的占位符 (%s, {user}) 是否在翻译中丢失或损坏。
    - **硬编码字符串:** 扫描代码库，查找未被提取到资源文件中的、需要本地化的硬编码字符串。