智见 AI BLUEBOOK SERIES

AI AGENT EDITION
ZJBB-003

AI AGENT 技术与实践蓝皮书

WorkBuddy Harness

技术与实践指南

把 UI 上的按钮，翻译回磁盘上的文件与发给模型的请求——Harness 设计拆解

大鹏主编
智见 AI 出品

候选版 · 0.1.0
目标版本 WorkBuddy 5.3.13 (build 20fd9da5c9a0cac41feb61b133f7bce4fe183a89)
资料截至 2026-08-15

目录

CONTENTS · ZJBB-003

前言 ····· 5

第 1 章 从按钮到文件：一次对话的解剖 ····· 6

1.1 第一环：你的话加上它的记忆，一起出发 ····· 6

1.2 第二环：模型只做一件事 ····· 7

1.3 第三环：工具执行，风险拦截 ····· 7

1.4 第四环：留下痕迹 ····· 7

1.5 第五环：交付与沉淀 ····· 8

1.6 拼起来：一张全景图 ····· 8

1.7 这本书怎么读 ····· 9

第 2 章 Harness 心智模型：官方五层与本书文件地图 ··· 9

2.1 为什么需要「挽具」这个词 ····· 9

2.2 官方五层模型 ····· 10

2.3 把五层当索引用 ····· 11

2.4 五层×本书章节对照总表 ····· 11

2.5 一个容易混淆的区分：Harness 与模型 ····· 11

第 3 章 ~/.workbuddy 全景：SQLite 管调度，文件管 Harness ····· 12

3.1 动手：三分钟逛完数据目录 ····· 12

3.2 六个家族 ····· 12

3.3 关键判断：为什么 Harness 走文件，调度走数据库 ····· 13

3.4 两个实用推论 ····· 14

3.5 隐私视角：这个目录里躺着什么 ····· 14

3.6 地图交接 ····· 14

第 4 章 记忆系统：三层存储与五类内容 ····· 14

4.1 先看规则原文：一段随每次请求注入的说明书 ····· 14

4.2 存储位置与文件形态 ····· 15

4.3 治理规则：限额、蒸馏与覆盖 ····· 16

4.4 实战：三个高频问题的记忆侧答案 ····· 16

4.5 补充机制：会话检索与双轨对照 ····· 17

4.6 边界 ····· 17

第 5 章 系统提示词的拼装：模板、片段与插槽 ····· 18

5.1 先看结果：一次请求的提示词长什么样 ····· 18

5.2 动手：找到你机器上的模板 ····· 19

5.3 四种交互模式：换模式就是换片段组合 ····· 20

5.4 插槽：记忆和身份怎么进提示词 ····· 21

5.5 变量全表与模式矩阵 ····· 21

5.6 验证：提示词不落盘，但注入有痕迹 ····· 22

5.7 架构判断与使用启示 ····· 23

5.8 边界与未决 ····· 23

第 6 章 身份系统：SOUL、IDENTITY、USER 与专家人设的边界 ····· 24

6.1 三块界碑的分工 ····· 24

6.2 注入去向：身份进了哪次请求 ····· 24

6.3 事故与边界：身份撞上专家 ····· 25

6.4 语气风格：可插拔的表达层 ····· 25

6.5 实战：安全地定制人格 ····· 26

6.6 边界 ····· 26

第 7 章 专家解剖：一个带 UI 的 SubAgent 插件 ····· 27
7.1 解剖台：一位专家的完整文件清单 ····· 27
7.2 人设提示词：专家的大脑 ····· 28
7.3 运行时：接管是如何发生的 ····· 29
7.4 专家团：编排层的另一形态 ····· 30
7.5 另一位样本：带规则的专家 ····· 30
7.6 实战：装专家前检查三件事 ····· 30
7.7 边界 ····· 31

第 8 章 插件系统：五个市场、三层作用域与供应链 ···· 31
8.1 三件套：登记、注册表、开关 ····· 31
8.2 五个市场：一次生态考古 ····· 31
8.3 插件能注入什么：四类内容 ····· 32
8.4 运行数据与生命周期 ····· 33
8.5 登记条目长什么样 ····· 33
8.6 供应链视角：三问一习惯 ····· 34
8.7 边界 ····· 34

第 9 章 技能与规则：双层 SKILL.md 与 WORKBUDDY.md ····· 35
9.1 技能是什么：一段按需加载的方法论 ····· 35
9.2 双层目录：用户级与项目级 ····· 35
9.3 规则文件：项目的宪法 ····· 36
9.4 十五分钟：写你的第一个技能 ····· 37
9.5 边界 ····· 37

第 10 章 会话与证据：JSONL、traces 与审计日志 ···· 38
10.1 会话流水：事件的 ParentID 链 ····· 38
10.2 文件历史：改动前的快照 ····· 39
10.3 执行轨迹与审计：另外两本账 ····· 39
10.4 实战：两个高频排查 ····· 40
10.5 事件之外：providerData 里的运行时元数据 ····· 40
10.6 隐私交叉提示 ····· 40
10.7 边界 ····· 40

第 11 章 连接器与 MCP：工具边界、OAuth 与审批 ···· 41
11.1 两套接入体系 ····· 41
11.2 数据流向：连上之前想清楚 ····· 41
11.3 审批：拦在哪一步 ····· 42
11.4 实战：接入第一个 MCP ····· 42
11.5 边界 ····· 42

第 12 章 安全与隐私：注入面、明文凭证与沙箱 ····· 43
12.1 风险一：提示词注入——内容变成指令 ····· 43
12.2 风险二：本地明文凭证 ····· 43
12.3 风险三：文件系统——权限模式与沙箱 ····· 43
12.4 风险四：数据出域的完整地图 ····· 44
12.5 十分钟安全自查 ····· 45
12.6 边界 ····· 45

第 13 章 生态坐标：同源、兼容与产品矩阵 ····· 46
13.1 出身：从代码助手到通用助手 ····· 46
13.2 文件级的亲缘证据 ····· 46
13.3 产品矩阵中的位置 ····· 47
13.4 对使用者的三个推论 ····· 48
13.5 边界 ····· 48

第 14 章 实战：排查与第一次安全定制 ····· 48
14.1 五步排查法 ····· 48
14.2 第一次安全定制：完整流程 ····· 50
14.3 排查案例走一遍 ····· 50
14.4 高阶组合：三个实用模式 ····· 51
14.5 维护节奏 ····· 51
14.6 本章边界 ····· 51

第 15 章 复用路线：视频课、认证课与图书的切法 ····· 52
15.1 一节课的骨架：90 分钟讲透 Harness ····· 52
15.2 认证单元：智能体工程师课程的嵌入方案 ···· 52
15.3 图书写作的复用边界 ····· 53
15.4 教学资产清单 ····· 53
15.5 全书收束 ····· 53

附录 A · ~/.workbuddy 目录速查表 ····· 54
A.1 完整速查表 ····· 54

附录 B · 五步排查卡（打印版） ····· 55
B.1 打印版排查卡 ····· 55

附录 C · 术语表 ····· 56
C.1 术语总表 ····· 56

附录 D · 版本快照与取证方法 ····· 57
冻结版本 ····· 57
核对本书是否仍适用于你的版本 ····· 57
读者可复用的取证方法 ····· 57
证据分层的公开标注 ····· 58

术语表 ····· 58

参考资料 ····· 58
官方 ····· 58
媒体 ····· 58
社区（交叉验证用） ····· 59
本地实测 ····· 59

PREFACE · ZJBB-003

前言

版本：公开测试版候选 · `as_of 2026-08-15` · 适用 WorkBuddy 5.3.13（build 20fd9da5，macOS）

署名：大鹏主编 | 智见 AI 出品

这本书写一件简单的事：把 WorkBuddy 打开给你看。

WorkBuddy 是那种上手快得不像话的产品——装上就能用，按钮清楚，谁都指挥得动。但用得越久你越会撞上一堵看不见的墙：它有时记得你三周前的一句话，有时忘了你昨天定的规矩；装个插件能力暴涨，也偶尔「变笨」；专家中心里一百多个名字，分不清哪个真有用。墙的后面是这套产品的真实身体：一套提示词模板、三层记忆、五个插件市场、身份文件、技能库、会话流水——全部落在你电脑上一个安静的目录里。

这本书带你走进那个目录。方法只有一条：**看到什么行为，就去找实现它的文件**。UI 上的每个按钮，都能翻译成本地的一个文件或一个字段；每次对话，都是一次可以拆解的拼装过程。全书十五章按这个方法走完 WorkBuddy 的 Harness：从一次对话的五环拆解，到五层心智模型，到记忆、提示词、身份三大件，到专家、插件、技能三层扩展系统，到会话证据与安全边界，最后落到实战排查与教学复用。

三个写作承诺。**第一，全部机制描述有证据**：来自本机文件实测（版本与指纹已冻结）、官方已取证文档或注明属性的推断，每处标注来源，做不到的地方明说。**第二，主路径不需要编程背景**：会打开终端复制粘贴命令就够了，深层细节都框在技术框里。**第三，不教破坏**：本书讲机制与治理，不讲绕过与破解。

读法建议：只想用好产品的读者，走第 1—3、5、7、12、14 章主线；想深度定制的实践者，补全第 4、6、8、9、10 章；要讲课的人，第 13、15 章为你写。所有动手环节只读或可逆，放心跟做。

一个诚实的前提：这个产品平均两天发布一个版本。书里的路径、字段、机制会过时——这是拆解快速演进产品的宿命，我们用版本冻结与更新方法（附录 D）来对抗它。但拆解的方法不过时：五层模型、「UI → 文件 → 请求」的追问路径、证据意识。带着这三样，你在任何 Agent 产品里都不会迷路。

智见 AI 蓝皮书系列 · 第三册

大鹏 2026 年 8 月，深圳

CHAPTER 01 · ZJBB-003

第 1 章 从按钮到文件：一次对话的解剖

你在 WorkBuddy 里打了一句话：「帮我把这个文件夹里的照片按月份整理成相册，生成一份索引文档」。十几秒后它开始列文件、读图片、建目录、写文档，中途弹了两次确认框，几分钟后交付了一个文件夹和一份 HTML 索引。整个过程你按过的按钮不超过三个。

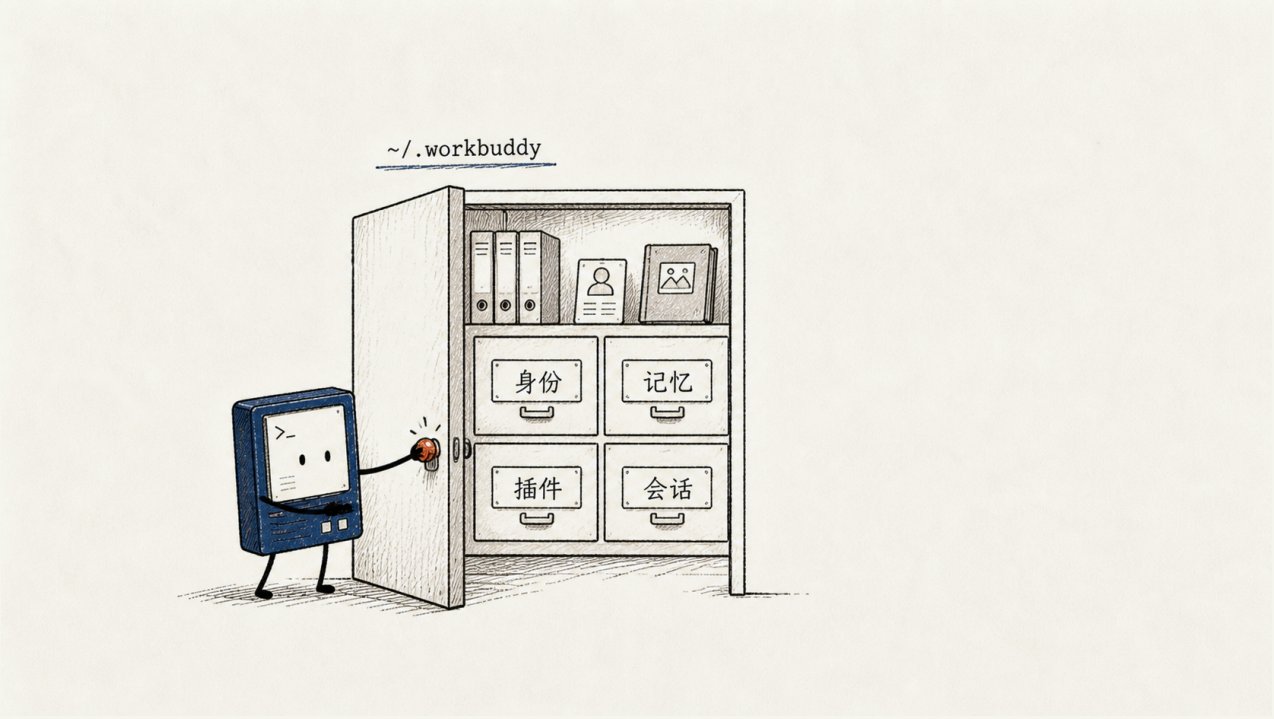
这是 WorkBuddy 想要的体验：把复杂性全部藏起来。但藏起来不等于不存在。这一章我们把这次对话放慢一千倍，看每一秒背后是什么在运转。这是全书的地基——后面十四章讲的每一个机制，都出现在这一次对话里。

读完本章，你应能把「WorkBuddy 帮我干活」这句模糊的话，拆成五个可指认的环节，并能说出每个环节在本地磁盘上的证据位置。

1.1 第一环：你的话加上它的记忆，一起出发

你按下回车的那一刻，发生的第一件事不是「发给大模型」。WorkBuddy 先组装上下文：你的这句话，加上系统提示词（第 5 章）、三层记忆里相关的内容（第 4 章）、当前项目的工作区信息（第 4、9 章），如果这次对话挂在某个专家身上，还有那个专家的人设提示词（第 7 章）。

这些零件几乎全部来自本地磁盘上一个你从没打开过的目录。打开 Finder，按 `Cmd+Shift+G`，输入 `~/.workbuddy`——从安装那天起，WorkBuddy 就把你的一切配置存在这里：身份、记忆、插件、技能、会话记录，几十上百兆的文件。**这本书要拆解的东西，物理上就是这个目录



WorkBuddy 的全部复杂性，都安静地躺在 ~/.workbuddy 这个目录里

图 1-1：WorkBuddy 的全部复杂性，都安静地躺在 ~/.workbuddy 这个目录里。

**。

1.2 第二环：模型只做一件事

上下文组装完成，请求发往模型。很多人对这一步有误解，以为 WorkBuddy「内置了一个很聪明的 AI」。更准确的说法是：模型收到一份很长的、结构化的请求，其中大部分篇幅是 WorkBuddy 拼进去的规则、记忆和工具说明书，你的那句话只占很小一块。

模型的角色像一位业务娴熟但完全不了解你的外援专家：它读规则、看工具清单、理解任务，然后决定下一步——通常是调用某个工具。模型不碰你的文件、不碰网络，它只输出「我想调用某某工具」的指令。真正动手的是下一环。

模型清单本身也是配置，不是写死的。社区解包研究在应用包内找到五个产品配置文件，声明了从混元、DeepSeek、Kimi、GLM 到 Claude 的几十个模型条目，每个条目标注能力（是否支持工具调用、图片、推理）与计费。你在界面上切换模型，改写的是请求发往哪里，而不是「换了一个 WorkBuddy」。

1.3 第三环：工具执行，风险拦截

模型说「我想执行 `mkdir 2026-07`」，指令落回到 WorkBuddy 的执行层。这里是安全机制的主场：这个操作在工作空间内、是新建而非删除，放行；下一秒模型想 `rm -rf` 某个目录，执行层拦下，弹确认框——就是你见过的那两次。

官方把这层叫默认权限模式：写敏感路径、删文件、跑脚本、访问网络四类操作要确认，其余放行；还有一档完全访问权限会关掉二次确认，官方文档明确列了不建议开启的场景（生产资料目录、桌面下载这类杂物目录、批量删除）。本地配置文件里能看到这层机制的落点：一份沙箱写路径白名单。第 12 章专门讲它。

工具执行完，结果连同报错一起回传给模型，进入下一轮。**这个「模型思考—调用工具—拿到结果—再思考」的循环，就是 Agent 与聊天机器人的分水岭**。聊天机器人一次回答完事；Agent 在循环里干活，一个任务转几十圈很正常。

1.4 第四环：留下痕迹

循环的每一拍都被记录。会话文件（JSONL 格式，按项目分目录存储）里能看到完整事件流：你的消息、模型的推理过程、每次工具调用的名称和参数、每次执行的结果，还有文件快照——每次 WorkBuddy 改你的文件之前，先存一份备份。另有执行轨迹（traces）目录记录执行链和 token 消耗，审计日志按天滚动。

这些记录不是给你看的，但你可以看，而且第 10 章会教你看。它们是这本书的证据来源，也是你排查问题的第一现场：「WorkBuddy 说它做了但文件没变」「昨天那个会话它到底改了什么」，答案全在里面。

1.5 第五环：交付与沉淀

任务结束，交付物落进工作空间。同时还在发生两件安静的事：这次会话的关键信息被写入记忆层（哪层、写了多少，受第 4 章讲的限额约束）；如果这是定时任务，下次执行时间写进本地数据库。工作没有真的结束，它只是转成了下次请求的一部分上下文。

1.6 拼起来：一张全景图

五环连起来，就是 WorkBuddy 的骨架：

```
你的输入
   ↓
[拼装] 身份+记忆+规则+工具清单+你的话 → 完整请求     ← 第4/5/6/9章
   ↓
[推理] 模型决策下一步                                  ← 模型配置
   ↓
[执行] 工具落地，风险拦截，文件备份                    ← 第12章
   ↓
[记录] 会话/轨迹/审计三路落盘                          ← 第10章
   ↓
[沉淀] 记忆更新，影响下一次拼装                        ← 第4章
   ↺ 回到推理，直到任务完成
```

每一环都有对应的本地文件，每一个「按钮」都落在图上的某个位置：插件按钮往拼装环节注入内容，专家按钮换掉人设插槽，Agents Team 按钮唤起编排工具，记忆开关控制沉淀环节。**

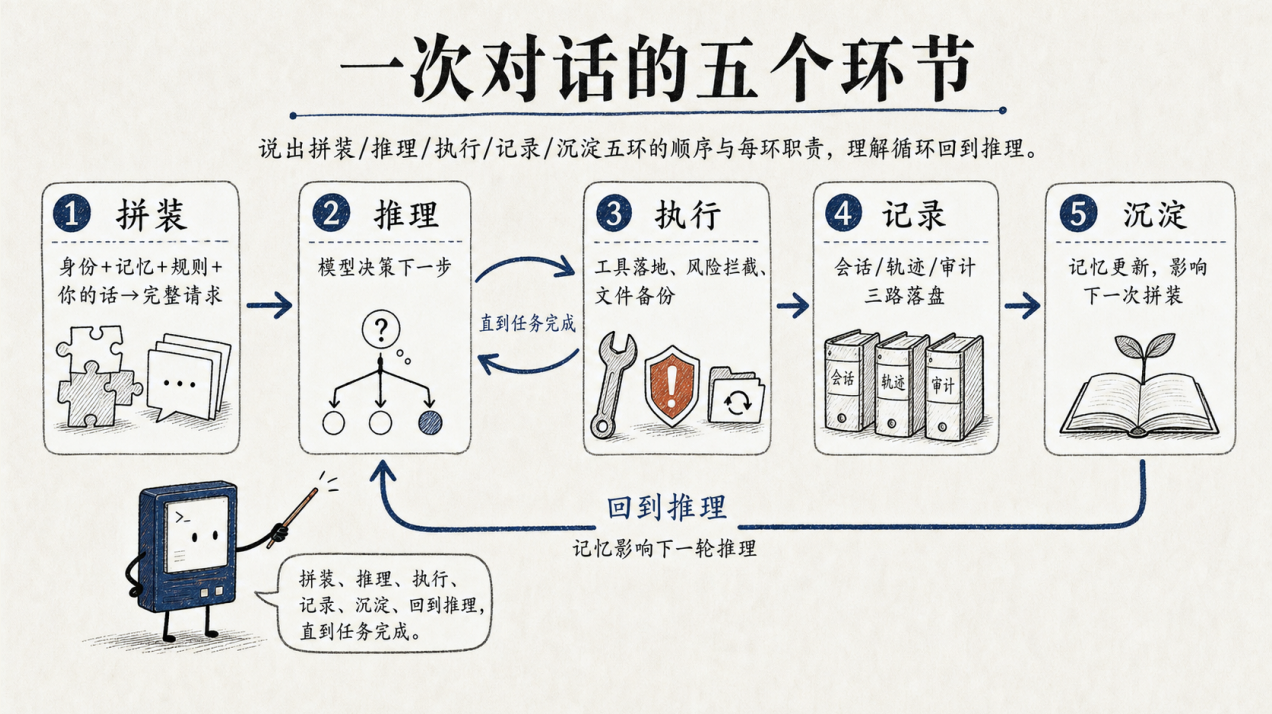


图 1-2：一次请求的五个环节与循环。

UI 上的每个控件，都是这张图某个节点的入口**——这是全书反复回到的一句话。

1.7 这本书怎么读

第 2 章建立 Harness 的完整心智模型，第 3 章带你走一遍本地目录。第 4 到 6 章讲「拼装」环节的三大件：记忆、提示词、身份。第 7 到 9 章讲扩展系统：专家、插件、技能。第 10 到 12 章讲证据与安全。第 13 章把 WorkBuddy 放回腾讯产品矩阵。第 14 章是实战：排查不生效、做第一次安全定制。第 15 章讲这套材料怎么变成课程。

只会按按钮的读者，主线读第 1—3、5、7、12、14 章；想动手定制的实践者，把第 4、6、8、9、10 章补全；要讲这门课的人，再加第 13、15 章的叙事层。

所有证据冻结于 WorkBuddy 5.3.13（2026 年 8 月）。这个产品平均两天发一版，书里的机制会有过时的一天——但拆解方法不会，那是你带得走的东西。

CHAPTER 02 · ZJBB-003

第 2 章 Harness 心智模型：官方五层与本书文件地图

2026 年，Agent 行业收敛出一个词：Harness（工程挽具）。 据 36 氪报道，OpenAI 在 2026 年初提出 Harness Engineering 的说法；36 氪对 WorkBuddy 团队的访谈里有一段适合当全书题眼的引文——WorkBuddy 负责人在解释 Red Hat 的对比时说：「sandbox 是 subtractive（减法式）的……Harness 是 additive（叠加式）的……Sandbox constrains，Harness enables.」沙盒约束，挽具赋能。你给模型套上的一整套让能力落地的工程系统，就是 Harness。

这一章建立全书的心智模型。它回答三个问题：Harness 到底包含什么？WorkBuddy 官方怎么分层？这套分层怎么落到你能摸到的文件上？

读完本章，你应能说出五层 Harness 各自的职责，并把任意一个 WorkBuddy 功能（插件、专家、记忆、定时任务）放进对应的层。

2.1 为什么需要「挽具」这个词

先解决概念问题。模型本身的能力边界和它实际完成任务的能力，中间隔着一整层工程。同一个模型，接上一套设计良好的工具、上下文管理和风险拦截，能稳定交付；裸着用，连「读一下本地文件」都做不到。这个差距就是 Harness 填的。

对普通用户，这个词的价值是把你已经感知到的现象命名。「WorkBuddy 比网页版 ChatGPT 能干活」——你现在可以说：两者模型能力可能相当，差距在 Harness。「装了某插件后 WorkBuddy 变笨了」——那是 Harness 的拼装层被塞进了低质量内容。命名之后，问题才可分析。

WorkBuddy 团队的工程复盘长文给出了官方的五层划分。以下每层的职责描述来自该文，本地文件对应列是本书的实测补充——这张对照表是全书的总纲。

2.2 官方五层模型

第一层：运行环境层——Agent 在哪里执行。 文件系统、Shell、沙箱、浏览器、MCP 与连接器、权限边界。这一层决定 Agent 的手脚能伸到哪里。本地对应：沙箱配置与写路径白名单（settings.json）、连接器目录、自带的语言运行时（binaries/ 下的 node 和 python）。

第二层：引导层——Agent 开始前掌握什么。 项目上下文、环境信息、规则与风格、技能、提示词缓存结构。任务开始前给足前置条件，模型就不用「盲人摸象」式探索。本地对应：系统提示词模板（第 5 章）、身份四件套（第 6 章）、技能目录（第 9 章）、规则文件。

第三层：反馈层——执行后如何获知错误。 可纠正的工具返回、时间戳校验、lint 与测试信号、审计日志。Agent 不怕报错，怕的是得不到报错。本地对应：会话文件里的工具结果事件、按天滚动的审计日志、文件历史快照。

第四层：编排层——多个能力如何组织。 渐进式加载、意图识别、多模型路由、团队协作、并行调用。当任务超过单循环能力，这层负责拆与合。本地对应：专家团工具入口（第 7 章）、技能的按需加载机制（第 9 章）、产品配置里的模型路由。

第五层：迭代层——Harness 自身如何演进。 这层不在磁盘上，在团队的迭代节奏里：随模型能力涨落精简或加码约束。据 36 氪报道，WorkBuddy 三个多月发了 43 个版本，平均不到两天一版；更新日志里「专家提示词不再注入身份文件」「工作空间自动创建记忆文件」这类条目，就是 Harness 在自我调整的证据。本书冻结的 5.3.13，是这条河流的一帧。



官方五层 Harness 与本地证据对照

图 2-1：官方五层 Harness 与本地证据对照。

2.3 把五层当索引用

五层模型的用法不是背诵，是当检索框架。遇到任何 WorkBuddy 行为，先问它属于哪层：

- 「它怎么会记得我上周说过的话」→ 引导层（记忆注入，第 4 章）。
- 「为什么删文件要确认」→ 运行环境层（权限模式，第 12 章）。
- 「专家团和普通对话有什么区别」→ 编排层（第 7 章）。
- 「它说改了但我找不到改动」→ 反馈层（会话与审计，第 10 章）。
- 「升级后行为变了」→ 迭代层（版本 diff，附录 D 的取证方法）。

这个框架也解释了本书为什么按现在的顺序组织：第 3 章先给全景地图，然后从引导层（记忆、提示词、身份）讲到编排层（专家、插件、技能），再回到运行环境层与反馈层（安全、证据），最后实战。

2.4 五层×本书章节对照总表

把官方五层、本地证据与本书章节放进一张总表，作为全书导航的收束：

Harness 层	职责一句话	本地证据位置	本书章节
运行环境层	手脚伸到哪	沙箱白名单、权限审批、连接器、自带运行时	第 11、12 章
引导层	开始前知道什么	提示词模板、身份四件套、技能、规则文件	第 4、5、6、9 章
反馈层	做错了怎么知道	工具报错回传、审计日志、文件快照	第 10 章
编排层	多能力怎么组织	专家与专家团、渐进加载、模型路由	第 7、8 章
迭代层	系统自己怎么进化	更新日志的机制条目时间线	附录 D、第 13 章 13.1

这张表也是读后自测的工具：合上书，任选一层，你能说出它的三个本地文件证据吗？说不出的那层，就是值得重读的部分。

2.5 一个容易混淆的区分：Harness 与模型

最后一个心智校准。用户报告「WorkBuddy 不好用」时，原因可能在模型（能力不足）、也可能在 Harness（上下文拼装不当、工具配置错、记忆污染）。两者的修复路径完全不同：前者换模型或拆解任务，后者改配置、清记忆、调插件。

判断归属的第一步，是看证据（第 10 章的方法）：会话文件里模型是否拿到了正确的上下文？工具调用是否被拦截？回答质量差但输入正确，多半是模型问题；该有的记忆没出现在请求里，是 Harness 问题。**把「AI 不行」翻译成「哪一层不行」，是从用户变成驾驭者的分界线。**

本书余下部分都在这张地图上行动。下一章，我们推开 `~/.workbuddy` 的大门。

CHAPTER 03 · ZJBB-003

第 3 章 ~/.workbuddy 全景：SQLite 管调度，文件管 Harness

第 1 章说过，这本书拆解的对象物理上是你的数据目录。现在正式开门。这一章的目标很实际：逛完 `~/.workbuddy`，给每个目录一个职责标签，让你之后任何时候想知道「某东西存在哪」，都能自己找到答案。

读完本章，你应能独立回答三个问题：WorkBuddy 在我电脑上存了哪些东西？哪些是配置、哪些是记录、哪些是敏感数据？为什么 Harness 资产几乎全是文件而不是数据库？

3.1 动手：三分钟逛完数据目录

打开终端，执行（全部只读）：

```
ls ~/.workbuddy/
```

你会看到四十多个条目。别慌，它们分六个家族。挨个看过去：

```
cat ~/.workbuddy/MEMORY.md
ls ~/.workbuddy/plugins/marketplaces/
ls ~/.workbuddy/skills/ | head
ls ~/.workbuddy/projects/
```

第一条读的是你的用户级长期记忆（内容可能只有寥寥几行，这正常，见第 4 章）；第二条列出插件市场（本机实测五个；你的机器装得少会更少）；第三条是你的技能库；第四条最有意思——projects 目录下的每个子目录名，就是你某个项目文件夹的绝对路径转写。找找看，你正在用的项目大概率躺在里面，装着你的全部会话记录。

3.2 六个家族

家族一：身份与记忆。 `SOUL.md`（人格）、`IDENTITY.md`（身份卡）、`USER.md`（你的画像）、`MEMORY.md`（长期记忆）四个文件，加上 `memory/`（云端记忆的本地缓存）和 `memery/`（一个拼写独特的目录，第 13 章讲它的来历）。这是第 4、6 章的主场。

家族二：扩展系统。 `plugins/`（插件本体与登记）、`skills/`（技能库）、`skills-marketplace/`（技能市场）、`connectors/` 与 `connectors-marketplace/`（连接器）、`extensions/`。判断「装了什么」看这里，第 7—9、11 章的主场。

家族三：配置。 `settings.json`（总开关：插件启停、通道、沙箱白名单）、`mcp.json`（MCP 服务器）、`models.json`（模型端点，含密钥，敏感）。第 12 章会讲这些文件的安全属性。

家族四：会话与证据。 `projects/`（按项目分组的会话 JSONL）、`traces/`（执行轨迹）、`audit-log/`（审计日志）、`file-history/`（文件改动快照）、`blobs/`（产物存储）、`artifact-index/`（产物索引）、`workspace/`（每会话工作区）。第 10 章的主场。

家族五：调度。 `workbuddy.db`——唯一的 SQLite 数据库，外加 `tasks/`、`plans/`、`todos/`、`automations` 相关表。存的是定时任务、会话统计、工作空间登记这类「运行时要查询的状态」。

家族六：运行时杂项。 `app/`（Electron 浏览器内核数据）、`binaries/`（自带的 node 与 python）、`logs/`、`shell-snapshots/`、缓存与迁移目录。出问题时 `logs/` 是第一现场。



数据目录六家族地图

图 3-1：数据目录六家族地图。

3.3 关键判断：为什么 Harness 走文件，调度走数据库

实测 `workbuddy.db` 的表结构，只有八张表：automations（定时任务定义）、automation_runs（执行历史）、automation_runtime_state（下次执行时间）、automation_delivery_outbox（投递队列）、sessions、session_usage（用量统计）、workspaces、迁移元数据。**没有记忆表、没有插件表、没有提示词表**——Harness 资产一个都不进数据库。

这是一个值得停下来想的架构判断。提示词模板、身份文件、记忆、技能、插件清单全部用纯文本文件（Markdown、JSON、YAML）承载，好处至少有三个：用户可以直接看和改（这本书才可能存在）；生态格式可以直接复用（WorkBuddy 的技能格式与开发者工具一致，第 13 章）；版本更新按文件分发（插件市场把市场当 zip 包下载解压）。数据库只承载需要高频查询和事务的调度状态。「**文件是给人和生态的，数据库是给调度器的**」——记住这个分工，你就能预测任何新功能落在哪。

3.4 两个实用推论

推论一：**备份与迁移变得简单**。想让另一台电脑长成这台的样式，理论上拷贝身份四件套、settings.json、skills、plugins 目录即可（敏感凭证另说，第 12 章讲风险）。官方的跨设备同步走它自己的通道，但文件在那儿，主动权就在你手里。

推论二：「**不生效**」**排查有了物理路径**。改了身份文件没变化？看文件是否真被读（第 5 章的拼装清单）、有没有被缓存、版本是否已变（市场自动更新可能改掉你以为稳定的东西）。第 14 章把这套排查整理成五步法。

3.5 隐私视角：这个目录里躺着什么

冷静看一下：你的身份画像、长期记忆、全部会话记录（含你发过的每句话）、文件改动历史、使用统计，都在这一个目录里，明文为主。这是能力的代价。三个动作降低风险：敏感对话后清理对应会话文件（第 10 章教你定位）；理解 models.json 里第三方模型端点意味着对话可能离开腾讯基础设施（第 12 章）；备份这个目录时按敏感数据对待。更完整的清单在第 12 章。

3.6 地图交接

附录 A 有完整的目录速查表，打印出来贴在手边。至此全书地图交付完毕：第 1 章的逻辑图、第 2 章的五层模型、本章的物理目录。三者互相垂直——逻辑图讲时间顺序，五层模型讲职责分工，目录地图讲物理位置。后面章节不再重复全景，直接进入各机制纵深。下一站：记忆系统，那个决定「它为什么记得你」的机制。

CHAPTER 04 · ZJBB-003

第 4 章 记忆系统：三层存储与五类内容

「它记得我」。这是用户对 WorkBuddy 最常报告的惊讶，也是最多的抱怨来源——记住了不该记的，忘了该记的，或者记的东西开始干扰新任务。这一章把「记忆」这个词拆开：WorkBuddy 的记忆不是一个东西，是三层存储、五类内容、一套治理规则（限额、蒸馏、覆盖）的总和。理解了结构，「为什么它记得/为什么它乱记」就都有了答案。

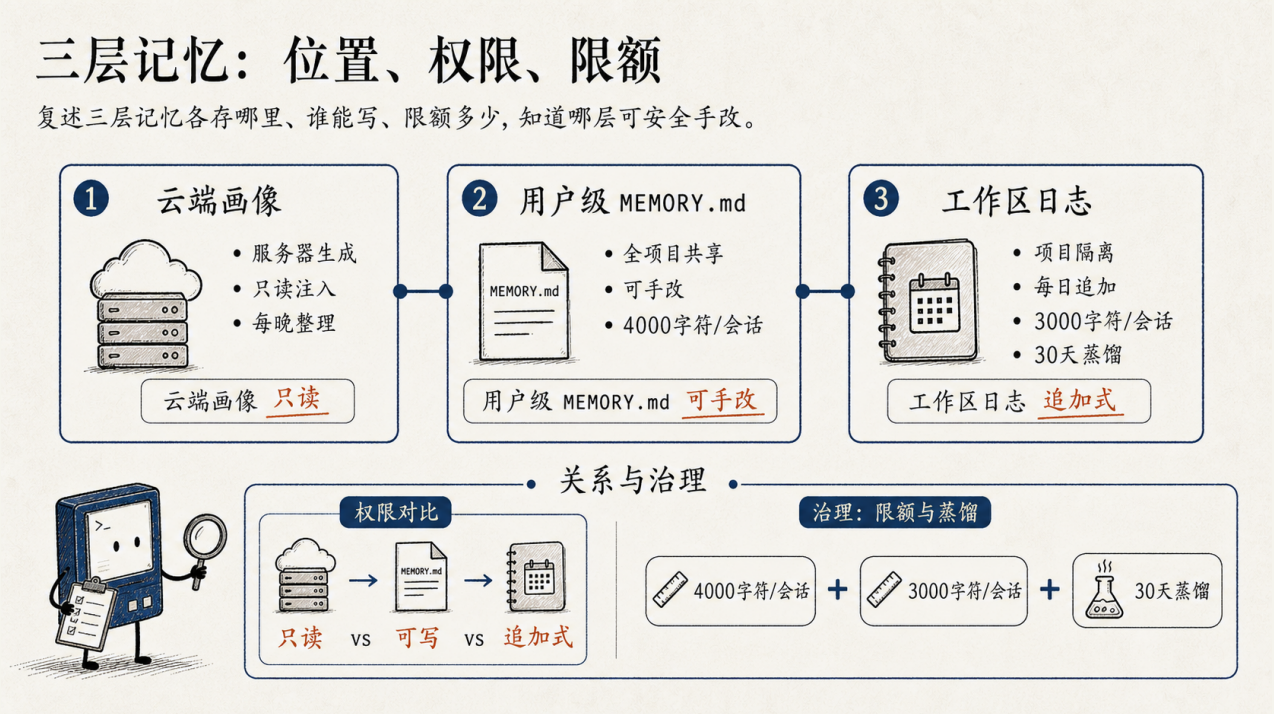
读完本章，你应能说出三层记忆各自的位置、读写权限和限额，知道什么内容会进哪层，并能动手查看和清理自己的记忆。

4.1 先看规则原文：一段随每次请求注入的说明书

WorkBuddy 自己的记忆机制说明，写在系统提示词的公共片段里（第 5 章讲过拼装机制）。这段文件的存在本身就很说明问题——**记忆系统不只是存储，还包括一份发给模型的使用规则**。规则要点如下（据本机 5.3.13 文件归纳）：

三层结构，权限各不相同。第一层云端记忆分两半：一半是服务器生成的用户画像，会话开始时注入，本地只读——文件缓存在 `~/.workbuddy/memory/`，你手动改了也会被下次同步覆盖；另一半是历史会话检索工具，需要时向服务器发起搜索，不占请求上下文。第二层是用户级本地记忆，就是那个 `MEMORY.md` 文件：全项目共享，写入限额每次会话四千字符，由模型在你明确说「记住这个」时更新。第三层是工作区记忆，只在当前项目生效：每天一个日期文件（追加式，不许改历史），加一份精炼的长期笔记（限额三千字符）；超过三十天的日志要求按主题蒸馏进长期笔记后删除。

配套还有官方的内容分类学，来自工程复盘长文：五类记忆——稳定事实、知识背景、行为信号、表达偏好、会话延续。值得注意的不是这五类本身，而是一个刻意的设计：**程序性记忆（怎么做事情的方法论）被排除在长期记忆之外**。怎么做事归技能系统管（第 9 章），记忆只管「你是谁、你知道什么、你偏好什么」。两个系统泾渭分明，这是很多人配置失败的根源——把该写成技能的方法论塞进了记忆，结果每次召回都不完整。



三层记忆的位置、权限与限额

图 4-1：三层记忆的位置、权限与限额。

4.2 存储位置与文件形态

把三层落到磁盘上：

```
~/.workbuddy/
├── MEMORY.md                    ← 用户级本地记忆（第二层）
├── memory/
│   └── <用户ID>_memory.md       ← 云端画像的本地只读缓存（第一层A）
└── （各项目内）
    └── .workbuddy/memory/
        ├── 2026-08-15.md        ← 工作区每日日志（第三层）
        └── MEMORY.md            ← 工作区长期笔记（第三层）
```

动手看一眼自己的：

```
cat ~/.workbuddy/MEMORY.md
```

很多人的这个文件短得出奇——几条偏好而已。这不是坏了，是设计使然：云端的画像记忆承担了大头，本地文件只放「必须逐字遵守的硬规则」。官方文档对记忆的说明是：由模型自动从会话提取、注入系统提示词、每晚整理。

另有一个历史彩蛋：数据目录里还有个拼写成 `memery` 的目录，装着另一套记忆文件。它是另一条产品线的兼容组件（第 13 章讲这段生态渊源），平时不参与 WorkBuddy 主记忆循环，但它的存在解释了为什么有些老教程会提到两个记忆文件。

4.3 治理规则：限额、蒸馏与覆盖

三层各自的治理逻辑值得细看，因为「记忆失灵」大多栽在这里。

云端画像的治理在服务器侧：自动提取、每晚整理，你不可直接编辑。它最容易出现的问题是「画像过时」——你三个月前的项目偏好还在影响现在的对话。对策不是删文件（删了会重建），而是在对话里明确纠正，让新信号进画像。

用户级 MEMORY.md 的治理是限额制的：四千字符每次会话。这个设计的意图是逼记忆保持精炼——它是硬规则清单，不是笔记本。实践中它的失败模式是「写了但没被读」：模型更新了文件，但下次拼装时这段内容排在请求靠后位置，权重不足。对策是把最关键的规则放在文件开头，且保持每条一行。

工作区记忆的治理最有工程味：日志追加式（不许改写历史，保证据链），长期笔记限额，三十天强制蒸馏。这其实是一套小型的知识管理流水线——日志是流水，笔记是库存，蒸馏是盘点。**看懂这套流水线，你就看懂了所有 Agent 记忆系统的通用范式**。

4.4 实战：三个高频问题的记忆侧答案

「它总记得我不要的东西」：定位到具体层。问它「你对我的印象是什么」，回答里的内容多半来自云端画像；在对话里明确说「以后不要再用这个信息」，让纠正信号进画像。用户级文件里的过时条目直接编辑删除——这是三层里你唯一能安全手改的。

「我让它记住，它却忘了」：检查三个可能——写入超限（这次会话的四千字符用完了）；写进了工作区层但你在别的项目问（层级不匹配）；内容被判定为会话延续类（只在本会话有效，这是正确行为不是故障）。

「换个项目它就失忆了」：不是故障，是设计。工作区记忆按项目隔离，跨项目的是用户级和云端层。想要跨项目复用的偏好，说「记住，以后所有项目都……」引导它写入用户级文件。

4.5 补充机制：会话检索与双轨对照

第一层记忆还有一个容易漏掉的部件：历史会话检索工具。它不占请求上下文，模型需要时主动调用，由服务端在全量历史会话里做检索排序。使用时机的设计很克制——提示词片段里写明的典型场景是用户明确提到过去的某次讨论（「我们之前聊的那个方案」），且当前上下文里找不到。**记忆注入（画像常驻）与记忆检索（按需召回）是两套机制**：前者是每次请求的固定成本，后者是按使用的边际成本。理解这个区分，你就明白为什么「它有时记得有时不记得」——常驻的画像过时会失真，按需的检索措辞不当就召不回。想让过去的讨论可召回，当时就把关键结论写进对话明确语句里。

双轨对照也补一笔实测细节：memory 与 memery 两个目录下的文件名都带同一串用户标识，后缀分别是 `_memory.md` 与 `_memery.md`，各配一份伴随文件（备份或状态）。两套文件并存且互不引用——一个服务 WorkBuddy 自己的记忆循环，一个保留另一产品线的格式。这是生态兼容在存储层的直接物证，也是你排查「记忆怎么有两份」疑问的答案。

4.6 边界

本章机制的版本边界：三层结构与限额数字来自 5.3.13 本机模板文件，模板随市场更新可能调整；云端画像的提取与整理逻辑在服务器侧，本书只能观测其本地缓存与行为效果，不声称了解其内部实现；`memory/` 缓存文件含个人信息，书中示例均已脱敏，你分享截图时同理。

下一章往上走：记忆是通过什么管道进入请求的——系统提示词的拼装机制，全书机制部分的核心一章。

CHAPTER 05 · ZJBB-003

第 5 章 系统提示词的拼装：模板、片段与插槽

同一个 WorkBuddy，你在问答模式问「你是谁」，和你在资讯专家对话里问同样的问题，得到的答案是两个人。切一次模式，行为跟着变；选一个专家，人格整个换掉。多数人把这理解为「产品里内置了很多机器人」。真实机制要具体得多：WorkBuddy 每次发送请求前，都会用一套写在本地磁盘上的模板文件，把身份、记忆、模式约束和专家人设拼装成一份完整的系统提示词（system prompt）。这一章带你找到那套模板，读懂它的拼装规则，并验证这套机制确实在运行。

读完本章，你应能在自己的电脑上定位 WorkBuddy 的提示词模板目录，说出一次请求的提示词由哪几类零件拼成，并理解为什么「改了不生效」往往不是玄学，而是你改的东西根本不在拼装清单里。

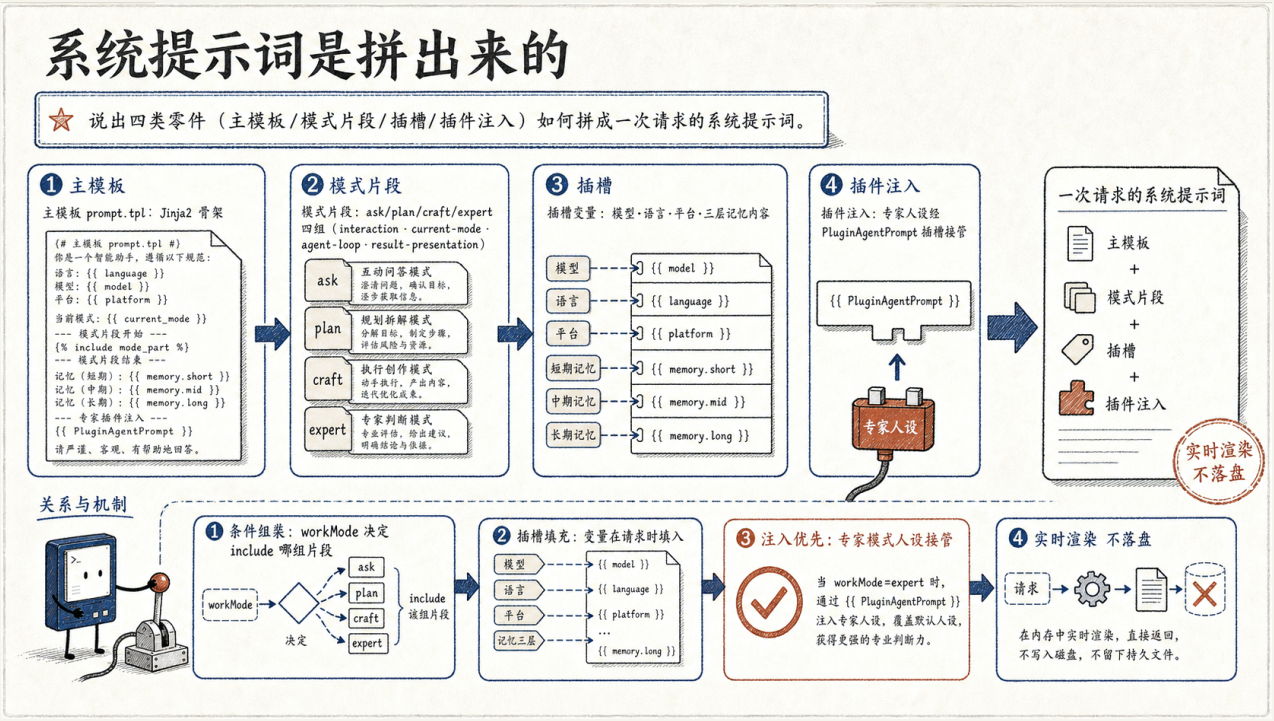
> 本章所有文件证据来自声明环境实测：macOS、WorkBuddy 5.3.13（build 20fd9da5），文件指纹以 SHA-256 冻结于本书「版本快照」。模板原文是腾讯的专有资产，本书只做机制分析并少量引用，完整文件请在本机查看。

5.1 先看结果：一次请求的提示词长什么样

把结论放在前面。在 5.3.13 的本地安装里，WorkBuddy 主线程的系统提示词由四类零件拼成：

1. **主模板**：一个带占位符的文本骨架，决定整份提示词的结构顺序；
2. **模式片段**：按当前交互模式（问答 Ask、规划 Plan、执行 Craft、专家 Expert）条件拼入的行为约束块；
3. **插槽变量**：模型名、界面语言、平台、三层记忆内容等运行时才确定的值；
4. **插件注入**：专家人设、规则文件等由插件系统提供的覆盖内容。

这不是官方文档写的，是你打开本地目录就能看到的事实。下面动手验证。



系统提示词的四类零件如何拼成一次请求

图 5-1：系统提示词的四类零件如何拼成一次请求。

5.2 动手：找到你机器上的模板

WorkBuddy 的数据目录在用户主目录下。打开终端，执行一条只读命令：

```
ls ~/.workbuddy/plugins/marketplaces/workbuddy-builtin/
```

你会看到几个名字直白的目录：`welcomemode`（欢迎模式）、`interactionmode`（交互模式）、`prompt-common`（公共提示词）、`builtin-plugins`（内置插件）。WorkBuddy 把自己的提示词资产做成一个「内置市场」里的插件包，和第三方插件走同一套格式。这个设计本身就有信息量：**产品自己的提示词和第三方扩展在文件层面是同一种东西**。

进入 `welcomemode/work/`，有一个 `prompt.tpl` 文件。用任何文本编辑器打开它——这就是主模板。开头几行长这样（引用自本机文件，占位符原样保留）：

```
This conversation is powered by {% if modelId == "fast-model" or modelId == "balanced-model" or modelId == "deep-model" %}Auto{% else %}{{ modelName }}{% endif %}
```

花括号是 Jinja2 模板语法的标志。`{{ modelName }}` 是变量插槽，运行时填入当前模型名；`{% if %}` 是条件块，遇到三个特定 modelId 枚举值时头部显示为 Auto——注意这三个标识只是模板层的显示分支，不是产品模型档位名（产品侧路由是 auto 智能路由加快速/深度两档）。 WorkBuddy 桌面客户端内部携带一套命令行引擎（社区解包研究称其在应用包的 `cli/` 目录），模板渲染就发生在每次请求前。

一个常见误解在这里顺便澄清：这份模板在应用安装包里也有一份（社区解包称 `resources/templates/`，约 19 个文件 3000 余行），插件市场目录里的这份会随市场自动更新。两个位置是同一模板体系的分发渠道，你本机运行时以市场目录为准。**自己动手改 `prompt.tpl` 并不推荐**——市场更新会覆盖你的修改，而且这些文件属于产品资产；用户可定制的层在规则文件和技能（见第 9 章），不在主模板。

5.3 四种交互模式：换模式就是换片段组合

`interactionmode/` 下有四个目录：`ask`、`plan`、`craft`、`expert`。每个目录里是同样名字的五个片段文件：

片段	负责什么
`interaction.md`	该模式下的交互总则与工具白名单
`current-mode.md`	告诉模型「你现在处于什么模式」
`agent-loop.md`	单步循环的行为规则（何时思考、何时调工具、何时收尾）
`result-presentation.md`	结果怎么呈现（格式、详略、交付物形态）
`tool-use.md`	工具使用的细则

主模板里用条件 include 把它们织进来。`prompt.tpl` 中反复出现这样的结构（简化引用）：

```
{% if workMode == "ask" %}{% include "interactionmode-ask/fragments/interaction.md" %}
{% elif workMode == "plan" %}{% include "interactionmode-plan/fragments/interaction.md" %}
{% elif workMode == "expert" %}{% include "interactionmode-expert/fragments/interaction.md" %}
{% else %}{% include "interactionmode-craft/fragments/interaction.md" %}{% endif %}
```

也就是说，你在界面上点一次「切换到问答模式」，落到文件层面就是：下次渲染系统提示词时，`workMode` 变量取值 `ask`，拼进去的是 `ask` 目录那一组片段。**模式不是不同的机器人，是同一骨架上换了一组约束**。官方工程复盘把这层能力归入「引导层」（feedforward）——在任务开始前给 Agent 正确的前置条件。

专家模式有一个特别的细节值得单独讲。`interactionmode/expert/fragments/current-mode.md` 的全文只有一行（Jinja2 注释包裹的一个插槽）：

```
{# {{ PluginAgentPrompt }} #}
```

当前模式的说明不再由内置片段提供，而是留了一个插槽，等插件系统把所选专家的人设提示词填进来。专家机制的完整链条在第 7 章展开，这里先记住结论：**「专家」接管主线程的入口，就在系统提示词的拼装层**。

再看 `interaction.md` 的开头，专家模式的这个文件带着一份 frontmatter 工具白名单（节选）：

```
tools:
  - Read
  - Write
  - Edit
  - Bash
  - WebFetch
  - WebSearch
  - Skill
  - Agent
```

```
  - Defer(TeamCreate)
  - Defer(TeamDelete)
```

工具列表里有两种写法：直接列出的（如 `Bash`）是默认可用；`Defer(...)` 包裹的（如 `Defer(TeamCreate)`）按需延迟加载——用到时才真正挂载。`TeamCreate` 和 `TeamDelete` 是专家团（Agents Team）的创建与解散入口，UI 上的团队协作按钮，底层就是这对工具。

5.4 插槽：记忆和身份怎么进提示词

模板里的变量不止模式和模型。实测枚举出的主要插槽包括：`ResponseLanguage`（界面语言）、`dataFolderName`（数据目录名）、`IsWindows`（平台分支）、`WorkingMemoryContent`、`UserLocalMemoryContent`、`UserMemoryContent`（三层记忆内容）等。

记忆插槽的注入点在公共片段 `prompt-common/fragments/memory-context.md`，全文就是三个变量并排：

```
{{ WorkingMemoryContent }}
{{ UserLocalMemoryContent }}
{{ UserMemoryContent }}
```

这三个插槽对应第 4 章讲过的三层记忆：工作区记忆、用户级本地记忆、云端用户档案。配套的 `workbuddy-memory-system.md` 片段则告诉模型这套记忆怎么用——哪层只读、哪层可写、写入限额多少。换句话说，**记忆系统不只是「存了数据」，还包括一份随请求注入的使用说明书**。

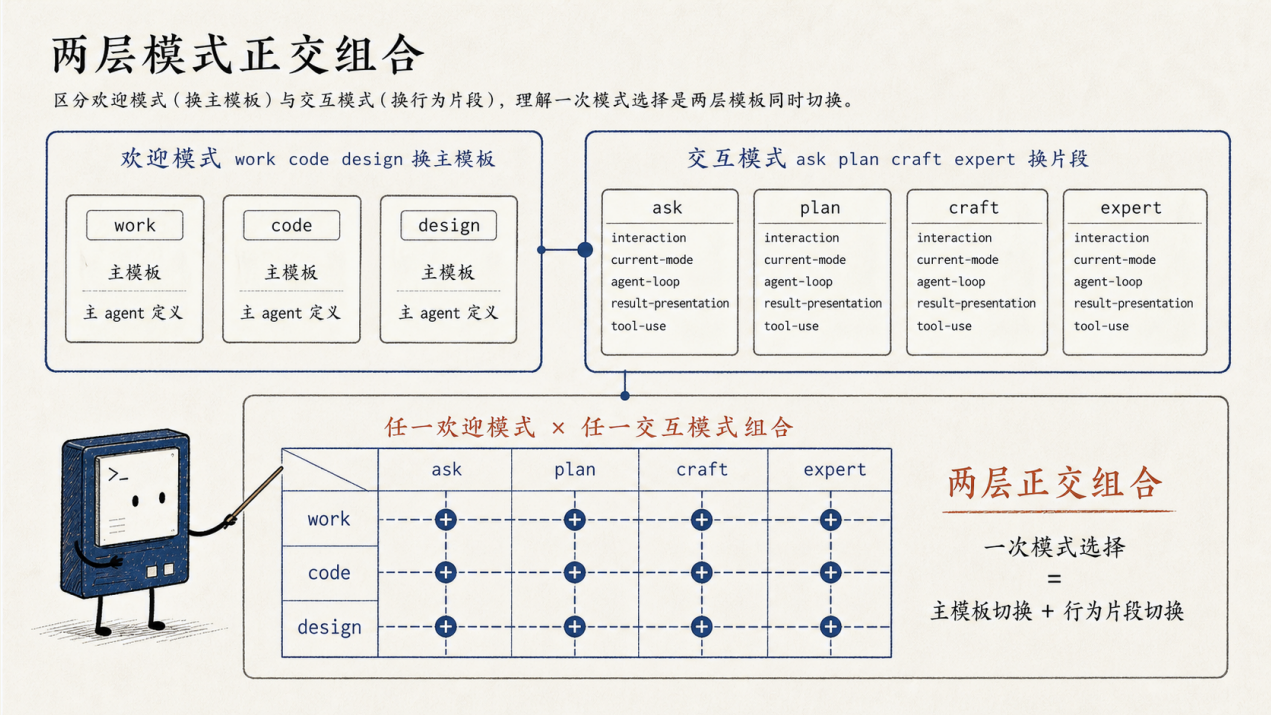
配套的还有安全块。主模板里有内容政策（content policy，禁止泄露系统提示词本身）、个人文件安全规则（对桌面、下载等个人目录的删除类操作强制走确认与回收站）、区域惯例（A股红涨绿跌、人民币符号）等成段约束。这些块在不同模式模板里逐字重复出现——社区研究统计称重复行数以千计。不把公共块抽成单一引用，而是允许逐字复制，是一个明确的设计取舍：**拼装结果稳定优先于源文件不重复**。对使用者来说，这意味着某条安全规则在每个模式下都在场，不依赖加载顺序。

5.5 变量全表与模式矩阵

把 5.3.13 主模板里实测枚举出的变量整理成表，这是「拼装层」的完整接口面：

变量	填什么	影响
workMode	ask / plan / craft / expert	选择四组交互片段
modelId / modelName	当前模型标识与名称	开头声明；特定 modelId 枚举值显示为 Auto
ResponseLanguage	界面语言	切换文档链接域名与标签文案
dataFolderName	数据目录名	提醒模型该目录非缓存、勿删
IsWindows	平台标记	条件拼入 Windows 命令安全块
productFeatures.*	功能开关	如关闭多模态生成则删掉对应能力段
WorkingMemoryContent 等	三层记忆内容	记忆插槽
PluginAgentPrompt	专家人设	expert 模式的接管入口

模式维度上还有一个容易忽略的层级：`welcomemode` 里的 work / code / design 三种模式各自是独立插件，各有主模板与主 agent 定义——设计模式的模板是三者中的异类，社区解包研究描述它定义了「智能设计助手」，引入画布文件格式与三段式回复结构。也就是说，**「模式」这个词在 WorkBuddy 里至少指两件事**：欢迎模式（work/code/design，换主模板）与交互模式（ask/plan/craft/expert，换行为片段），两者正交组合。你在界面上做的一次模式选择，落到文件层可能是两层模板的同时切换。



欢迎模式与交互模式正交组合

图 5-2：欢迎模式与交互模式正交组合。

5.6 验证：提示词不落盘，但注入有痕迹

读完模板你可能想问：能不能看到拼装后的最终提示词？在会话记录里找不到。WorkBuddy 的会话文件（按项目存储的 JSONL）只记录用户消息、模型回复、推理过程和工具调用事件，没有任何 system 类型的条目——**系统提示词每次请求时实时渲染，用完不落盘**。

但注入的痕迹留在两个地方。其一，专家会话的首条用户消息以系统提醒（system reminder）通道开头。在声明环境中复现到这样一个样本：选择「数字生命卡兹克」资讯专家后发起对话，会话文件首条用户消息的结构是：

```
<system-reminder>
<expert_selection state="active">
The user has selected the "数字生命卡兹克" expert...
（完整粘贴专家插件的 agents/aihot.md，含 frontmatter，约 1800 字符）
</expert_selection>
</system-reminder>
（用户的实际问题）
```

专家插件的人设文件被完整注入，模型的后续推理原话是「根据专家指令，我必须调用 aihot 技能获取实时数据，不能凭训练数据猜测」——与专家提示词里的强制条款逐字对应。**人设不只被塞进了上下文，还实际支配了行为**。

其二，执行轨迹（traces）目录记录每次请求的执行链：生成（generation）与工具调用交替的跨度序列、token 消耗计数。它证明拼装后的请求真实发生，但同样不记录提示词内容。证据链到此闭合：模板文件证明「怎么拼」，会话注入痕迹证明「拼了什么进去」，轨迹证明「确实发了」。

5.7 架构判断与使用启示

把证据收拢成三个判断。

判断一：提示词是产品，不是咒语。 模板、片段、插槽分层，模式之间共享骨架、差异集中在约束块——这是工程化管理的提示词资产，有版本、有分发渠道（插件市场）、有功能开关（模板里的条件变量）。理解这一点，「AI 表现不稳定」的讨论就多了一个维度：同一产品不同版本、不同模式、不同插件组合下，系统提示词并不相同。

判断二：你的定制层不在主模板里。 用户能安全触碰的是规则文件、技能和身份四件套（分别见第 9、6 章），它们通过插槽和注入通道进入拼装流程；主模板和安全块是产品保留层，改了也会被更新覆盖。「为什么不生效」的第一排查点由此而来：先确认你改的东西，到底在不在这次请求的拼装清单里。

判断三：system-reminder 是通用注入通道。 专家人设、插件规则（第 8 章的 `alwaysApply` 规则文件）都走同一个通道进入对话。这既是扩展性的来源，也是供应链风险的入口——装一个插件，等于允许它的规则进入你后续的每次对话。评估插件时的第一个问题应该是：它会注入什么？

5.8 边界与未决

本章证据的边界：模板内容冻结于 5.3.13，随市场更新可能变化；「桌面客户端内嵌 CLI 引擎」的判断来自社区解包研究与本地运行时痕迹的互证，官方未公开对应架构文档；提示词渲染后的完整全文在客户端内不落盘，本书以「模板 + 注入痕迹 + 执行轨迹」三层证据替代，未声称持有渲染后全文。

下一章往上走一层：被插槽引入的身份四件套（其中的 MEMORY 已在第 4 章）——SOUL、IDENTITY、USER——以及它们和专家人设擦枪走火的那个版本事故。

CHAPTER 06 · ZJBB-003

第 6 章 身份系统：SOUL、IDENTITY、USER 与专家人设的边界

数据目录根部躺着四个文件，文件名全大写，像四块界碑：SOUL.md、IDENTITY.md、USER.md、MEMORY.md。第 4 章讲了最后一块（记忆），这一章讲前三块——WorkBuddy 的身份系统。它决定你的助手「是谁」：性格、口吻、使命，以及它眼里的你是谁。

这个系统值得单独一章，因为它是一次真实产品事故的主角：某个版本里，用户自定义的身份内容会覆盖专家人设，导致「专家不像专家」。修复那次事故的更新，恰好把身份系统的边界暴露得干干净净。

读完本章，你应能说出三个身份文件各自的分工与注入去向，理解身份与专家人设的冲突机制，并能安全地定制自己的助手人格。

6.1 三块界碑的分工

SOUL.md（灵魂）： 人格与工作方式（以下示例为脱敏改写）。写的是「它怎么做事」——性格特质（真诚、可靠、冷静）、工作方式（先自助探索再提问、能派子任务就派）、表达口吻（短句、结论先行、禁用空洞过渡）。这是一份行为宪法，粒度到「禁用什么表达」。

IDENTITY.md（身份卡）： 它是谁。名字、角色定位、背景、气质，甚至表情符号。像一张员工工牌加上自我介绍。

USER.md（用户画像）： 你是谁。称呼、城市、时区、职业、工作方式、核心价值观，以及「与他协作的注意事项」。这份文件让助手理解对面的人，与 SOUL（助手自己的人格）互为镜像。

三个文件在安装引导（onboarding）阶段生成，之后你随时可改。格式全部是普通 Markdown——**身份在这套系统里不是模型参数，是磁盘上的文本**，这是它能被定制、被迁移、也被误用的根源。

这组文件名不是 WorkBuddy 独创。它们来自同一公司另一条开源产品线的约定（第 13 章展开），WorkBuddy 兼容了这套格式。你在界面上做的「个性化设置」，最终落盘就是这三个文件的修改；反过来，直接编辑文件同样生效——文件是唯一的真相源。

6.2 注入去向：身份进了哪次请求

按第 5 章的拼装模型，身份文件属于引导层资产，在请求组装时进入系统提示词（具体插槽位置与记忆相邻）。需要特别注意的是注入的身份文件会影响所有常规对话——work、code、design 模式下，你的助手始终带着这份人格出场。

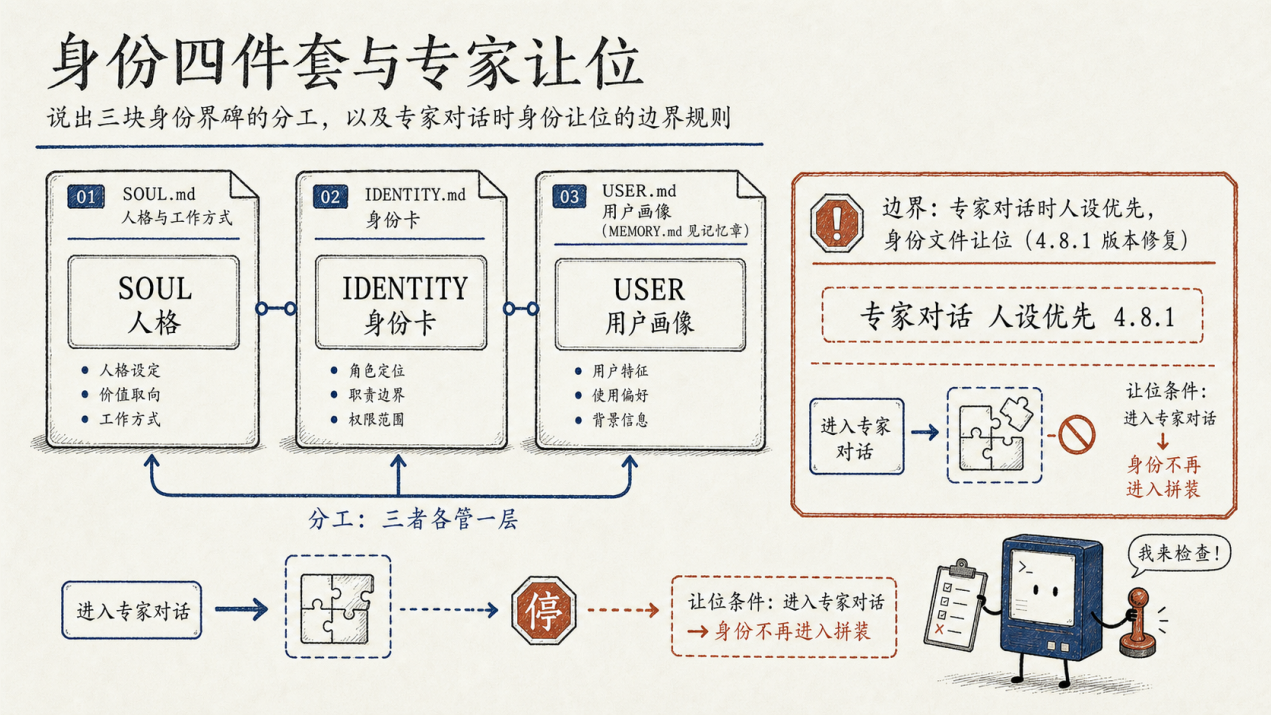
社区解包研究在安装包内的模板里发现了独立的「身份注入」辅助模板，以及一组风格文件（职业、创意、友善、直率等七种语气可选）。也就是说，**人格由三块界碑决定基调，语气风格是运行时可插拔的插件**——你在界面上切换的「语气」，落到文件层就是换了一段风格文本。

6.3 事故与边界：身份撞上专家

现在讲那次事故。WorkBuddy 的专家（第 7 章）各自带完整人设：选「数字生命卡兹克」资讯专家，对话里就该是卡兹克的口吻。问题在于：如果用户在身份文件里把自己的助手捏成了「小张」，而系统提示词又注入了这份身份内容，两个「我是谁」就在同一次请求里打架。由此可以推断，4.8.1 之前的版本正是这种状态——身份内容进入了专家对话的拼装，专家人设被稀释；官方修复条目反向证实了这条注入路径的存在。

官方更新日志 4.8.1 的修复条目原文：「不再注入 IDENTITY.md 内容，确保专家人设不被覆盖」。一行字，划出了一条重要的架构边界：**身份系统管辖常规助手；专家对话时，专家插件的人设优先，用户身份让位**。本机 5.3.13 的专家模式模板佐证了这个设计——专家模式下当前模式说明是一个空插槽，等待专家插件的内容填入（第 5 章 5.3 节），身份文件不再进入拼装清单。

这次事故的教学价值超过事故本身。它演示了 Harness 设计的经典难题：多个「人格来源」叠加时，优先级必须在拼装层显式裁决，否则模型收到矛盾指令，行为随机。你在定制任何 Agent 时都会遇到同款问题：公司规范、项目规则、个人偏好、临时指令，谁压谁？WorkBuddy 的答案是一个明确的让位规则，简单但有效。



身份四件套分工与专家对话时的让位边界

图 6-1：身份四件套分工与专家对话时的让位边界。

6.4 语气风格：可插拔的表达层

身份之外还有一层更轻的个性化：语气。社区解包研究在安装包模板里发现了一组风格文件——职业、创意、友善、直率、高效、苏格拉底式等七种，每种一份 Markdown，按用户选择拼进请求。它与三块界碑的关系是「宪法与穿着」：SOUL 决定这个人怎么做事，风格文件决定他这天穿什么衣服出门。

这个分层的设计价值在于解耦。人格重塑是高成本高影响操作（改 SOUL 要备份、要观察多轮）；语气调整是低成本操作（界面上切一下，落盘只是换了注入的风格段）。产品把高频低风险的个性化做成开关，把低频高风险的做成文件——**改动的便利程度，应该与它的影响半径成反比**，这个交互设计原则在 Harness 的每一层都成立，你后面设计自己的规则与技能体系时可以直接借用。

还有个实用观察：身份文件的修改在旧会话不生效（拼装发生在请求时），但语气切换几乎即时可感——因为它影响的正是当前对话的表达层。改了身份没变化时，先确认你开的是不是新会话。

6.5 实战：安全地定制人格

想动手的读者，三条路径按侵入度排序。

路径一（零风险）：界面个性化。 在设置里调整称呼、语气。适合改「它怎么说话」。

路径二（低风险）：编辑 USER.md。 用任何编辑器打开 `~/.workbuddy/USER.md`，修正画像里过时或错误的内容——这是四个文件里最值得手工维护的一个，因为画像质量直接决定它对你的理解精度。改完开新会话生效（拼装发生在请求时，旧会话不追溯）。

路径三（中风险）：重塑 SOUL.md。 写人格宪法是高杠杆操作：写得好，助手处处顺手；写得差，处处别扭。两个实践要点：用「正面陈述」而不是禁令清单（「用短句直接给结论」优于「不要啰嗦」）；控制篇幅，宪法越长，单条约束的执行力越弱。改之前备份原文件，不满意随时回滚——这是文件式身份的红利。

一条安全提醒：身份文件会随请求发送给模型服务方。不要在 SOUL 或 USER 里写 secrets（密码、密钥、身份证号）——它们不是保险箱，是每次请求的组成部分。

6.6 边界

本章边界：三个文件的注入细节（拼接位置、是否全模式注入）以 5.3.13 模板与更新日志为证；4.8.1 之前的具体冲突表现未在本机复现（本机版本已是修复后），事故叙述基于官方更新条目与机制推断，标注为推断；风格文件的七种语气清单来自社区解包，未逐个验证效果。

下一章进入全书最像「产品魔法」的部分：专家系统。你会看到那位「数字生命卡兹克」，在磁盘上到底是什么样子。

CHAPTER 07 · ZJBB-003

第 7 章 专家解剖：一个带 UI 的 SubAgent 插件

专家中心是 WorkBuddy 最像应用商店的地方：满列专家排成列表（产品模板自述 100 余位），每人有头像、职业标签、一句简介，点进去还能看到三个快捷问题按钮。选一位「资讯速递专家」，对话框里的助手立刻换了人格、换了能力、换了说话方式。

这一章做一次完整解剖。解剖对象是本机安装的一位真实专家——「数字生命卡兹克」（AI 资讯速递）。我们会把它拆到文件级，看清「一位专家」在磁盘上的完整形态，然后观察它在运行时如何接管对话。结论先给：**专家就是一个插件包，头像、开场白、快捷按钮全部是插件清单里的字段；对话开始时，专家的人设提示词注入主线程，替代通用助手人格。**

读完本章，你应能说出专家插件的四件套结构、UI 元素与清单字段的对应关系，理解专家与专家团的区别，并知道装一位专家前该检查什么。

7.1 解剖台：一位专家的完整文件清单

在专家市场目录下，每位专家一个文件夹。卡兹克专家的全部家当是四个文件：

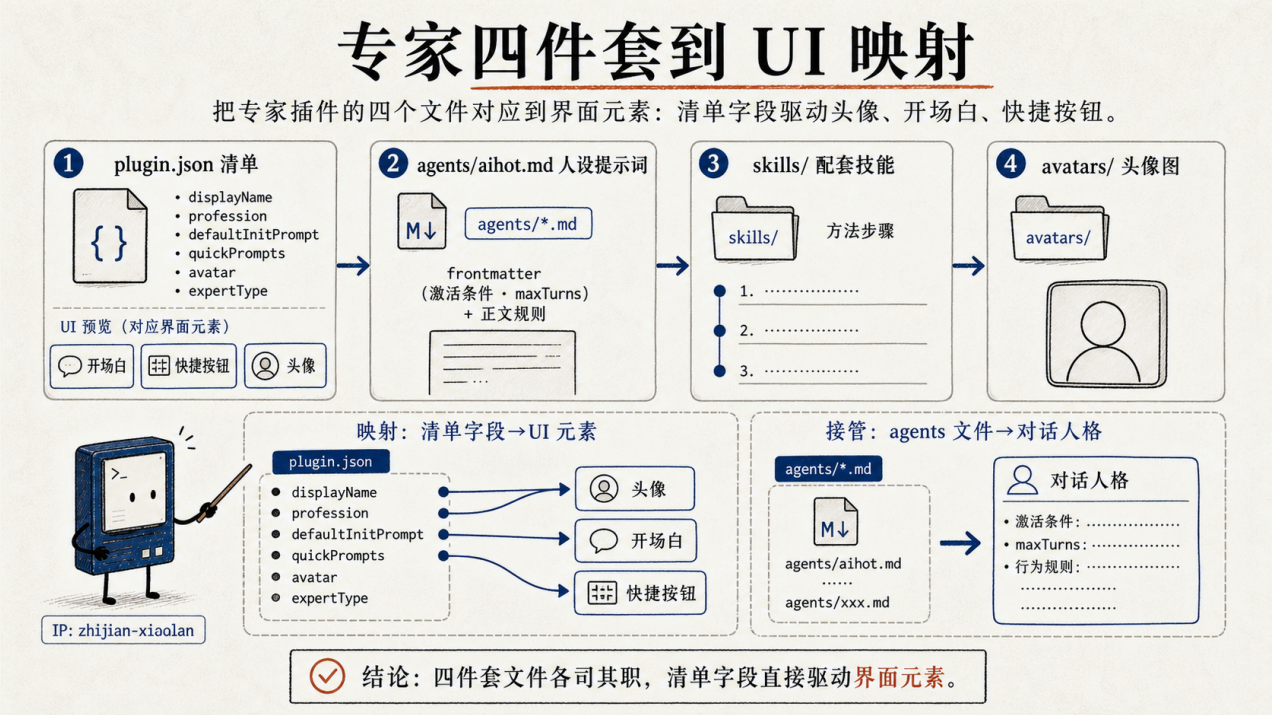
```
aihot/
├── .codebuddy-plugin/plugin.json   ← 插件清单（manifest）
├── agents/aihot.md                 ← 人设提示词（核心）
├── skills/aihot/SKILL.md           ← 配套技能（干活方法）
└── avatars/expert.png              ← 头像
```

先看清单。`plugin.json` 里除了名称、版本、作者，有一组专家特有字段（节选自本机文件）：

```
{
  "name": "aihot",
  "agents": ["./agents/aihot.md"],
  "skills": ["./skills/aihot"],
  "expertType": "agent",
  "displayName": { "zh": "数字生命卡兹克", "en": "Kazik" },
  "profession": { "zh": "资讯速递专家", "en": "AI News Briefing Expert" },
  "defaultInitPrompt": { "zh": "今天 AI 圈有什么新动态？" },
  "quickPrompts": [
    { "zh": "今天 AI 圈有什么新动态？" },
    { "zh": "看一下最新的 AI 日报" },
    { "zh": "OpenAI 最近发布了什么？" }
  ],
  "avatar": "avatars/expert.png"
}
```

对照你在界面上看到的：头像 = `avatar` 字段指到的图片；职业标签 = `profession`；进入对话时预填的那句问话 = `defaultInitPrompt`；对话框上方的三个快捷按钮 = `quickPrompts` 数组。**专家中心的每一格 UI，都能在这份清单里找到字段级对应。**这就是「UI 按钮 → 本地文件」翻译法最完整的一个样本。

注意 `expertType: "agent"`——它声明这位专家是「单 Agent 型」。还存在另一种类型：Team 型（专家团），稍后讲。



专家四件套与 UI 字段映射

图 7-1：专家四件套与 UI 字段映射。

7.2 人设提示词：专家的大脑

`agents/aihot.md` 是专家的本体，一份完整的系统提示词。结构分两段。开头是 frontmatter（YAML 元数据）：

```
---
name: aihot
description: AI 资讯速递专家，实时查询每天精选的 AI 动态。当用户提到 AI 资讯、AI 日报、AI 热点……时激活。
maxTurns: 30
---
```

`description` 不只是介绍——它写明了激活条件，什么话题会唤起这位专家；`maxTurns: 30` 限制这位专家单次对话最多转三十圈，防止失控。正文则是给模型的完整人设与工作规则：你是谁、核心能力五项、输出规范（默认生成 HTML 简报，给出配色、布局、分区要求；备选 Markdown 格式的规则）。

最有分量的是中间那段「技能调用声明」，原文风格是：收到 AI 资讯问题时**必须**调用配套技能获取实时数据，**不得**凭训练数据猜测当前动态。一份禁令加一份路由表（用户意图 → 技能里的哪个工作流）。这类强制条款的效果，马上会在运行时观测里看到。

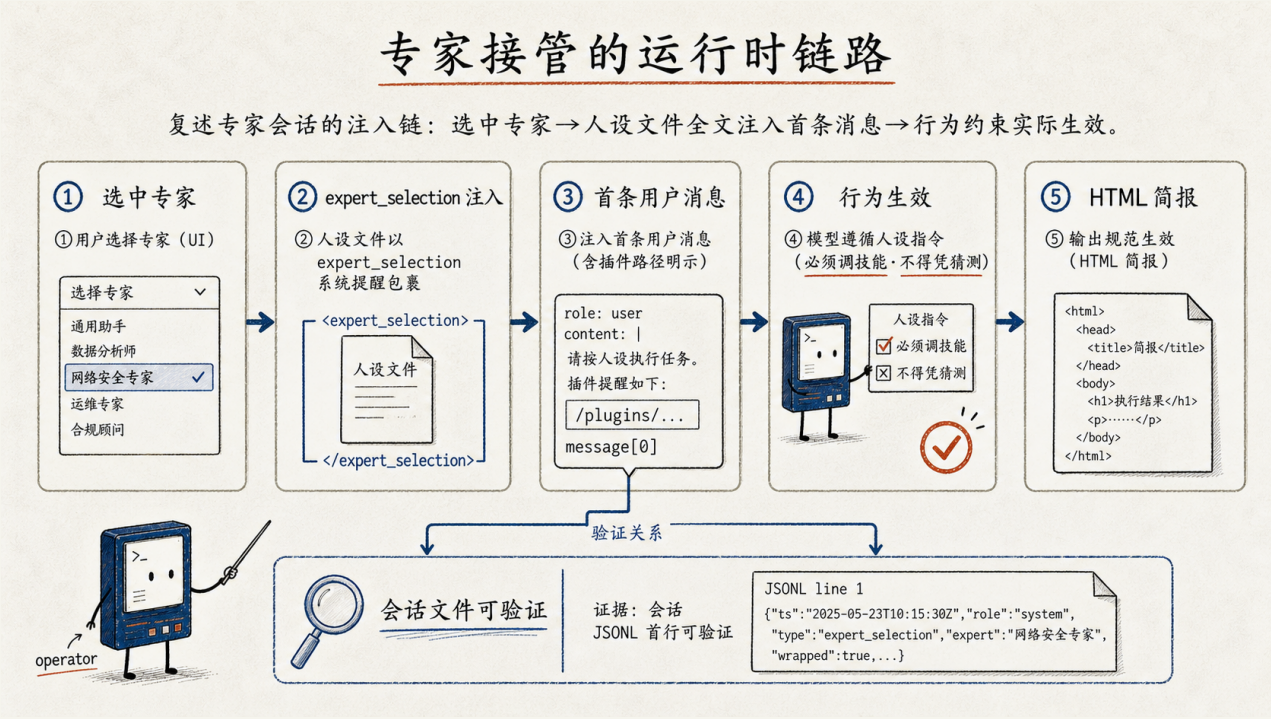
7.3 运行时：接管是如何发生的

第 5 章讲过专家模式模板里那个 `{{ PluginAgentPrompt }}` 插槽，现在看它运行时被填充后的样子。在声明环境中做了一次受控实验：选择卡兹克专家，发送「今天 AI 圈有什么新动态」。会话文件的首条用户消息结构如下（已脱敏缩略）：

```
<system-reminder>
<expert_selection state="active">
The user has selected the "数字生命卡兹克" expert...
Expert plugin path: ~/.workbuddy/plugins/marketplaces/experts/plugins/aihot/agents/aihot.md
（此处完整粘贴 agents/aihot.md 全文，含 frontmatter）
</expert_selection>
</system-reminder>
今天 AI 圈有什么新动态?
```

专家的人设文件被系统提醒通道完整注入。接下来模型的行为链值得逐帧看：推理记录原话是「根据专家指令，我必须调用 aihot 技能获取实时数据，不能凭训练数据猜测」——**人设里的强制条款直接改写了模型的行为决策**；随后它先找到并阅读技能文件，按里面的命令模板调用数据接口，拿到返回后按人设里的输出规范生成了 HTML 简报。

一次专家对话 = 清单字段渲染 UI + 人设注入 + 技能执行。没有魔法，全是文件。



专家接管的运行时链路

图 7-2：专家接管的运行时链路。

7.4 专家团：编排层的另一形态

专家中心里还有一类「专家团」：多位专家加一套协作流程。官方文档的定义对比很清楚——技能是能力，专家是能力加经验，专家团是多位专家加协作。落到本地形态：Team 型专家的插件里同样有清单与人设，但工作方式是团长拆解任务、分派成员、并行执行、整合交付（官方文档专家中心页描述其积分消耗约为普通对话的三到五倍，适合复杂任务）。

单机文件层面能看到的入口有两处：专家模式工具白名单里的 `Defer(TeamCreate)` 与 `Defer(TeamDelete)`（团队创建与解散的延迟加载工具），以及插件市场里的团队协作类插件。专家团的多轮协作过程发生在运行时编排层，本机未留存完整的团队会话样本——这部分标注为官方定义加文件入口证据，协作细节留作读者实验。

7.5 另一位样本：带规则的专家

本机另有一位已下载的视频生成专家，结构与卡兹克同构，但多出一个值得研究的目录：`rules/`。规则文件的元数据写着 `alwaysApply: true`，正文以系统提醒形式包裹一段场景声明（「用户已选择视频生成场景」），随后是严格的工作流约束——第一步必须用选择工具与用户确认画幅、帧率、时长，规格未确认前禁止执行任何后续步骤。

把它与卡兹克对照，能看出专家设计的两种配方：卡兹克把全部约束写进人设提示词（agents 文件），随对话注入一次；视频专家则把「人设」与「常驻规则」拆开——人设管身份与能力，规则管不可违反的流程纪律。后者更适合高风险、多步骤的专家（视频渲染失败成本高，所以规格确认被设计成硬门槛）。**装专家时看它有没有 rules 目录、写了什么，等于提前读到这位专家的「脾气」**。

两个样本也划出了专家市场的现实边界：市场里的内容由各路作者生产，格式统一但质量与安全水位不齐——这正是下一章插件供应链话题的入口。

7.6 实战：装专家前检查三件事

专家是第三方内容，装之前花两分钟做尽职调查。按侵入度查三样：

一查**注入面**。打开专家的 `agents/*.md`，通读人设：它要求模型必须做什么、禁止做什么？有没有要求访问外部地址、发送数据？这段文本将进入你后续每次与它的对话。

二查**能力面**。看清单里的 `skills` 与 `mcps`（如果有）：技能文件里有没有下载可执行内容、要求输入密钥的步骤？能力越强，越要确认来源可信。

三查**来源**。清单里的作者、版本、主页字段是否完整；市场里的发布者是否官方或可追溯。本机实测的专家样本来自个人作者，内容质量高，但这恰恰说明这个市场走的是社区生态模式——**信任决策在你，机制只负责分发**。

7.7 边界

本章边界：解剖样本为专家市场两位本机专家之一，样本量小，不声称覆盖全部专家形态（比如 Team 型专家的完整文件结构未取得样本）；「每一格 UI 对应清单字段」的映射基于 5.3.13 实测的字段集合；运行时证据来自单次受控会话，行为稳定性未做统计。

下一章从单个专家拉远到整个插件系统：市场怎么分发、插件怎么登记、规则怎么注入——以及为什么说插件是 Harness 供应链的入口。

CHAPTER 08 · ZJBB-003

第 8 章 插件系统：五个市场、三层作用域与供应链

专家是插件，但插件远不止专家。第 5 章的系统提示词模板装在「内置市场」里，文档处理能力来自「官方市场」，团队协作件来自「Teams 市场」——WorkBuddy 的一半能力是以插件形式存在的。这一章把插件系统整体摊开：市场怎么分发、安装怎么登记、插件能往你的对话里塞什么。

这一章的最后一个词是「供应链」。插件机制给了第三方进入你请求上下文的通道，能力与风险同源。读完本章，你应能看懂插件目录的三件套登记结构，说出五个市场各自的定位，理解插件的四类注入物，并建立装插件前的评估习惯。

8.1 三件套：登记、注册表、开关

WorkBuddy 的插件管理用三个文件协作，全在 `~/.workbuddy/plugins/` 下。

安装清单 `installed_plugins.json`：记录装了什么。每个条目的键名是「插件名@市场名」，值里有安装路径、版本、安装时间，还有一个值得注意的字段——`scope: "user"`。作用域字段的存在暗示插件可以装在不同层级（官方工程复盘明确说了三层：团队、项目、个人），用户级只是其中一层。

市场注册表 `known_marketplaces.json`：记录从哪装。本机注册的市场信息里能看到完整分发链路：市场以压缩包形式从官方下载域名分发，支持自动更新，包内一份清单列出市场里的全部插件及其技能路径。

总开关 `settings.json` **的** `enabledPlugins`：装了不等于启用。每个插件一行布尔值，你在界面上点开的每个开关，落盘就是这行 true 或 false。

三件套的分工是干净的注册-配置分离：市场注册表管「货源」，安装清单管「库存」，设置文件管「通电」。排查插件问题时按这个顺序查：先看开关，再看安装，最后看市场。

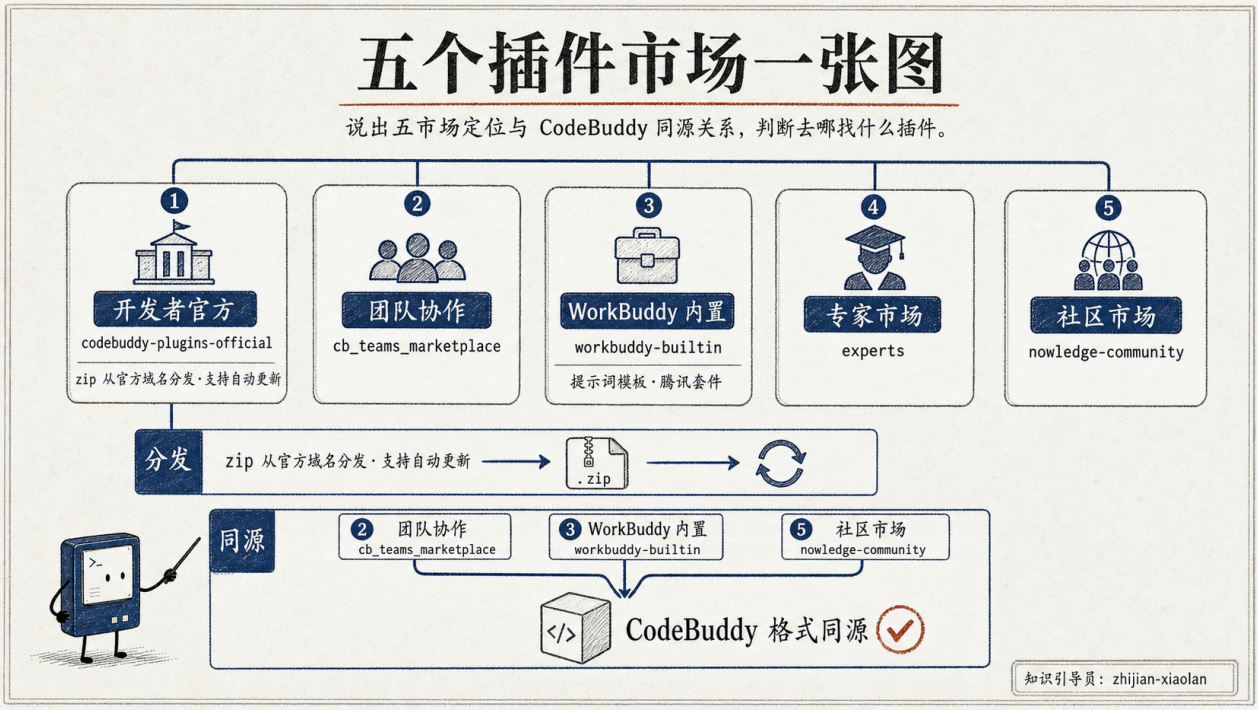
8.2 五个市场：一次生态考古

本机 5.3.13 实测有五个市场目录，名字本身就是一份生态地图：

市场	定位	本机样本
codebuddy-plugins-official	开发者官方插件市场	浏览器自动化、文档处理、LSP 语言服务、敏捷工作流

市场	定位	本机样本
cb_teams_marketplace	团队协作市场	内部通讯、文档技能、财务数据
workbuddy-builtin	WorkBuddy 内置	提示词模板、腾讯文档套件、支付、记忆片段
experts	专家市场	资讯专家、视频生成专家
nowledge-community	社区市场	知识管理类

注意前两个市场的名字：CodeBuddy 是腾讯的开发者工具品牌。WorkBuddy 的插件格式（`.codebuddy-plugin` 清单目录）、技能市场格式（`.codebuddy-skill`）、分发域名全部复用 CodeBuddy 体系——**WorkBuddy 的扩展生态不是从零建的，是站在开发者工具生态上长出来的**。这件事的完整含义在第 13 章展开，这里先记住它解释了一个实际好处：为 CodeBuddy 写的插件，理论上能直接在 WorkBuddy 里安装运行。



五个插件市场与同源关系

图 8-1：五个插件市场与同源关系。

8.3 插件能注入什么：四类内容

一个插件包可以向你的 WorkBuddy 注入四类东西，侵入度递增。

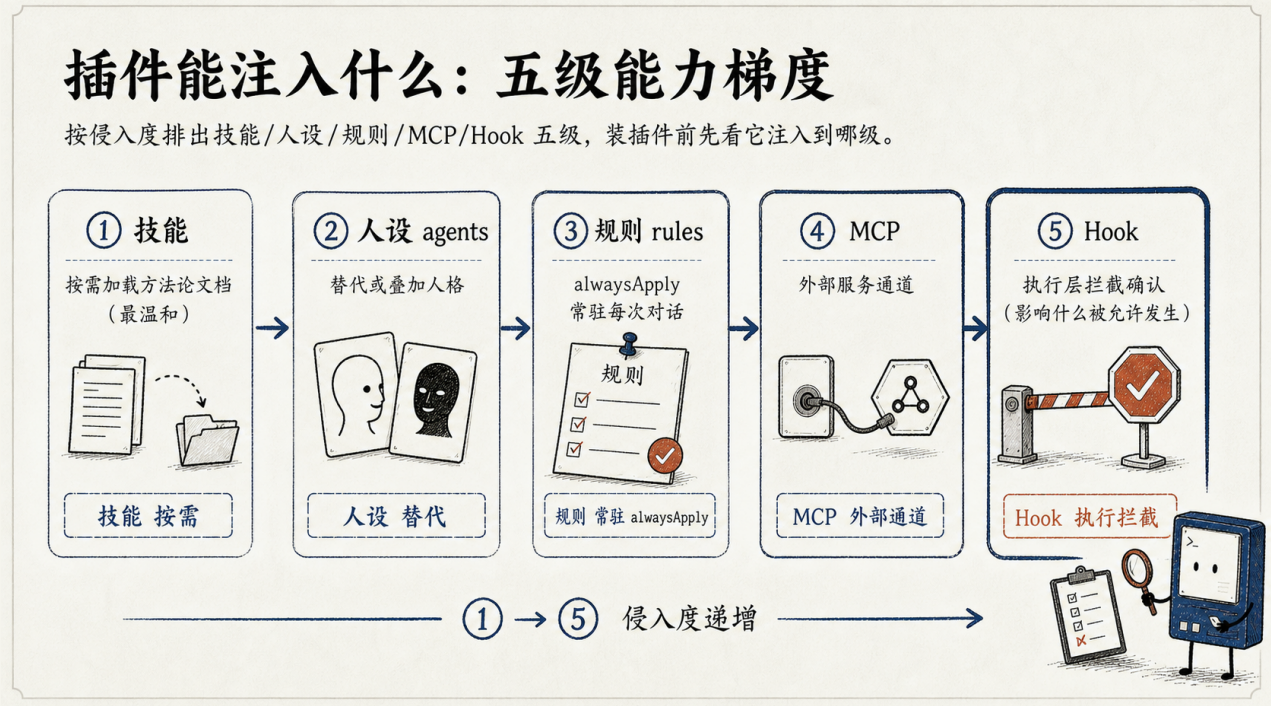
第一类：技能（skills）。 一套「怎么做某事」的方法论文档，模型按需读取。最温和——技能不进系统提示词，只在被调用时进入上下文。

第二类：智能体（agents）。 人设提示词，专家就是这类内容的典型。注入时替代或叠加默认人格（第 6、7 章的机制）。

第三类：规则（rules）。常驻约束。本机专家样本的规则文件带着这样的元数据：`alwaysApply: true`。翻译过来：**只要这个插件启用，这条规则进入你每一次对话**，不问你当时在干什么。插件规则还支持按场景注入——样本里能看到以系统提醒形式包裹的「用户已选择视频生成场景」声明，规则与场景绑定。

第四类：MCP 与工具（mcps）。直接连到外部服务的通道，第 11 章专讲。

四类内容里，规则是评估插件时最要紧的一环。技能是「书架上的书」，规则是「贴在显示器上的便条」——后者永远在场。装一个带 `alwaysApply` 规则的插件，等于接受它对你全部对话的持续影响。



插件五级注入能力的侵入度梯度

图 8-2：插件五级注入能力的侵入度梯度。

8.4 运行数据与生命周期

插件不只有代码，还有状态。`plugins/data/` 下每个插件一个运行数据目录（本机可见文档套件与支付插件的数据目录）；`plugins/cache/` 存各市场的已安装版本，市场更新走「下载新版到缓存、改安装指针」的路径。生命周期上，更新日志显示插件机制本身也在快速演进——5.3.5 版本条目「WorkBuddy 扩展插件 Hook」：插件可以在指定事件（如删除文件、发送消息、支付）上挂确认拦截。Hook 是四类注入之外的第五种能力：**不只影响说什么，还能影响什么被允许发生**。

8.5 登记条目长什么样

把安装清单的真实条目结构展开（本机样本，字段名为实测）：

```
"playwright-cli@codebuddy-plugins-official": [
  {
    "scope": "user",
    "installPath": "~/.workbuddy/plugins/cache/codebuddy-plugins-official/playwright-cli/
```

```
0.1.0",
    "version": "0.1.0",
    "installedAt": "2026-08-11T02:26:23.897Z",
    "lastUpdated": "2026-08-15T05:42:41.853Z"
  }
]
```

三个阅读点。键名里的 `@` 后缀是市场标识——同一个插件名可以同时存在于多个市场，装了哪个由键名决定。值是数组，为多版本、多作用域的并存留了结构空间（本机全部是单元素）。`lastUpdated` 与 `installedAt` 分开记，配合市场的 `autoUpdate` 标志，能看出这个插件是「装完就没动」还是「天天在更新」——排查行为突变时，先看这个时间戳。

上一节的钩子（Hook）机制在这里再强调一句安全面：恶意插件的 Hook 可以把高风险操作伪装成低风险。**评估插件时，有 Hook 的要看它拦什么、放什么。**

8.6 供应链视角：三问一习惯

插件系统本质是一条软件供应链：市场（货源）→ 清单（目录）→ 注入（交付）。供应链的安全逻辑在这里全部适用。

评估任何插件，三问：作者是谁、可不可追溯？注入物是什么——四类内容各有哪些，`alwaysApply` 的规则写了什么？能力边界在哪——要密钥吗、连外部服务吗、装 Hook 吗？

一个习惯：装完插件，翻一眼它落到你磁盘上的实际文件（`plugins/cache/` 对应目录）。清单声称的与文件实际的是否一致，是供应链诚信的基本检查。这不是偏执——第 12 章会讲到，本地配置里明文躺着的通道令牌和模型密钥意味着什么。

8.7 边界

本章边界：市场与登记结构为 5.3.13 本机实测；「三层作用域」来自官方工程复盘的自述，本机只观测到用户层实例，团队层与项目层的实际形态未取得样本；插件是否有签名校验机制未观测到证据，标注为未知——不声称有，也不声称没有。生态兼容性（CodeBuddy 插件直接可用）为格式同源推断，逐插件可用性需实测。

下一章讲插件四类注入物里最普及的两类：技能与规则文件——普通用户最可能亲手写的东西。

CHAPTER 09 · ZJBB-003

第 9 章 技能与规则：双层 SKILL.md 与 WORKBUDDY.md

技能（Skill）是 WorkBuddy 用户最先接触的定制手段，也是最被低估的。装一个「深度调研」技能，助手突然会做证据审查了；装一个「公众号排版」技能，产出直接符合栏目规范。规则文件则是它的沉默搭档：项目根目录放一份说明文件，助手在这个项目里的行为就变了。

这一章讲清两件事的文件形态与生效机制，并带你在十五分钟内写出第一个能用的技能。读完本章，你应能解释技能与规则的区别、双层技能目录的叠加关系，并完成一次安全的定制。

9.1 技能是什么：一段按需加载的方法论

先破除一个误解：技能不是代码。技能的主体是一份 Markdown 文档——SKILL.md，开头一段元数据（名字、一句「什么时候用我」的描述），正文写「这类任务怎么做」。模型在对话中判断当前任务匹配某个技能的描述时，读取该文档并遵循其中的方法。

这个设计的关键词是**按需加载**。系统提示词寸土寸金（第 5 章），技能清单只把「名字+描述」放进基础上下文——一行一个；完整方法论文档等你真的做这类任务时才读取。官方把这归入编排层的渐进式加载：用一份目录换一整座图书馆的按需取用。

动手看自己的技能库：

```
ls ~/.workbuddy/skills/ | head -20
head -8 ~/.workbuddy/skills/<某个技能名>/SKILL.md
```

第二条例子里你会看到每份技能的「门牌」：frontmatter 里的 name 和 description。**description 是技能的命脉**——模型凭它决定何时唤醒这个技能。写技能的一半功力在写好这一句话：具体、带触发词、划清边界（「用于……，不用于……」）。

9.2 双层目录：用户级与项目级

技能存在两个层级：用户级 `~/.workbuddy/skills/`（全项目可用）和项目级 `<项目>/.workbuddy/skills/`（仅当前项目）。实测本机一个真实项目：项目级 97 个技能是用户级 101 个的子集，同名文件内容完全一致——项目目录像是用户库在这个项目里的同步副本，加上项目特有的技能。

这个设计的意图是隔离：工作项目的技能不污染个人项目，反之亦然。给团队分发规范时，把技能放进项目目录随代码仓库走，每个成员打开项目即得同一套方法——这是把「组织知识」版本化的正确姿势。

优先级问题（同名技能两层都有时用谁的）本机未取得运行时样本，这里给读者留一个十五分钟的受控实验，顺带验证整章机制：

```
LABDIR=$(mktemp -d /tmp/skill-lab.XXXXXX)
mkdir -p "$LABDIR/.workbuddy/skills/demo-probe"
```

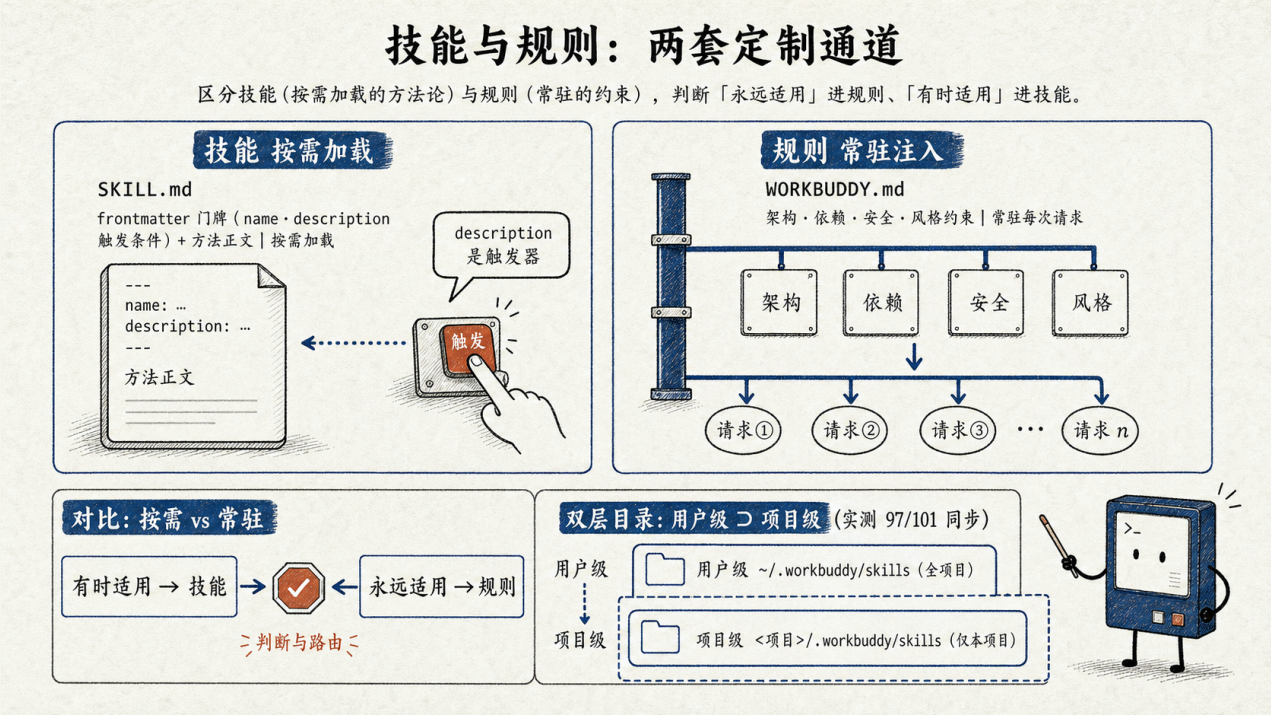
在 demo-probe 下写一个最小 SKILL.md（name 用一个你在用户级没有的词，description 写「当用户说『探针』时，回复『项目级技能已激活』」），在 WorkBuddy 里打开 $LABDIR 目录作为项目，说「探针」。然后把这个技能复制进用户级、改掉回复文案，再问一次——两次回答的差异就是优先级的直接证据。实验完删除临时目录即可。

9.3 规则文件：项目的宪法

技能管「怎么做某类任务」，规则管「在这个项目里永远遵守什么」。官方工程复盘提到的约定是：项目根目录放规则文件（社区与官方材料中出现的名字包括 WORKBUDDY.md，与开发者工具通用的 AGENTS.md 同族），记录架构约定、依赖边界、安全要求、代码风格；子目录可以放局部规则做更细的约束，原则是「项目规范放工作区规则文件」。

规则与技能的生效机制不同：**技能按需进入，规则常驻在场**。规则文件的内容在相关项目的每次请求里都会被注入（引导层资产）。这决定了两者的写作纪律——规则要短、要硬、要可执行（「提交信息用中文，格式：动词+对象」），技能可以长、可以软、可以讲道理（「做调研时优先找一手来源，理由是……」）。

实践中最常见的错误是拿规则当技能写：把一大段「如何做 X」塞进规则文件，结果每次请求都背着这段负重，模型反而抓不住重点。判断标准一句话：**「永远适用」进规则，「有时适用」进技能**。



技能（按需）与规则（常驻）两条定制通道

图 9-1：技能（按需）与规则（常驻）两条定制通道。

9.4 十五分钟：写你的第一个技能

现在写一个真正有用的。选一个你重复解释过三遍以上的偏好或流程——比如「周报格式」。在用户级建目录：

```
mkdir -p ~/.workbuddy/skills/weekly-report-style
```

写 `~/.workbuddy/skills/weekly-report-style/SKILL.md`：

```
---
name: weekly-report-style
description: 周报写作规范。当用户要求写周报、周总结、weekly report 时使用。不用于日记、月报或项目复盘。
---

# 周报规范

结构三段：本周结论（3 条以内）、数据变化（带数字）、下周计划（按优先级）。

规则：
- 每条一行，动词开头，不写套话
- 数字必须来自用户提供的材料，缺失就标注「待补」
- 总长度不超过 300 字
```

保存后开新会话说「帮我写周报」。如果它按三段结构出稿，链路就通了；如果没触发，九成是 description 不够明确——改那句触发条件，再试。这就是技能开发的完整循环：写门牌、写正文、实测、改门牌。

进阶方向：正文里可以引用同目录下的参考文档（子文件放模板、示例），技能目录是一个小知识包而不必是单文件；官方市场里的成熟技能（如文档处理套件）是最好的范文。

9.5 边界

本章边界：双层目录的同步关系基于单项目实测，优先级结论待读者实验补全（9.2 的受控实验即为此设计）；规则文件的名字与作用域约定来自官方工程复盘与社区材料，5.3.13 本机未预置示例文件；「按需加载」的内部触发机制（是模型自主判断还是框架拦截）不可观测，两种解释在社区材料中并存，本书按「描述匹配」的行为层理解。

至此，扩展系统的三层（专家、插件、技能）讲完。下一章换视角：不看你给了它什么，看它留下了什么——会话、轨迹与审计，Harness 的反馈层与你的证据现场。

CHAPTER 10 · ZJBB-003

第 10 章 会话与证据：JSONL、traces 与审计日志

前面九章讲机制怎么运转，这一章讲运转留下的记录。对普通用户，这些记录是「出问题去哪说理」的现场；对进阶用户，它们是理解与调优的数据源。WorkBuddy 在你的磁盘上同时记三本账：会话流水（JSONL）、执行轨迹（traces）、审计日志。三本账各记一面，合起来能回答关于「刚才到底发生了什么」的几乎一切问题。

读完本章，你应能用一条命令定位任意会话的记录文件，看懂 JSONL 里的事件类型，会查一次文件改动的历史，并知道三本账各自的边界——它们证明什么、不证明什么。

10.1 会话流水：事件的 ParentID 链

会话记录在 `projects/` 下按项目分组，目录名是项目路径的转写，每个会话一个 JSONL 文件（一种一行一个 JSON 对象的文本格式）。一行一个事件，实测的事件类型有六种：

事件类型	记录什么
message	用户消息与模型回复
reasoning	模型的思考过程
function_call	模型发起的工具调用（名称+参数）
function_call_result	工具执行结果（含报错）
file-history-snapshot	文件改动前的快照标记
ai-title	会话的自动命名

最有价值的设计是每个事件都带 `parentId`：指向上一个事件，串成链。这让「第 3 步的判断依据是什么」变得可追溯——顺着链往回走，能看到模型读到哪个文件、拿到什么结果、然后做了什么决定。**这条链就是 Agent 的「工作底稿」**，也是本书能做机制验证的根本原因（第 7 章的专家注入证据、第 5 章的「无 system 事件」结论都来自它）。

值得专门指出会话文件里**没有**的东西：系统消息。第 5 章讲过，系统提示词每次请求实时拼装、不落盘。所以会话文件记录的是「对话与行动」，不记录「规则全文」——想知道某次请求拼进了什么规则，回到模板与注入痕迹去推（第 5 章方法）。

动手，找你最近一次会话：

```
ls -lt ~/.workbuddy/projects/*/*.jsonl | head -3
```

`-lt` 按时间排序，前几个就是你最近的会话。看某个会话里模型调过哪些工具：

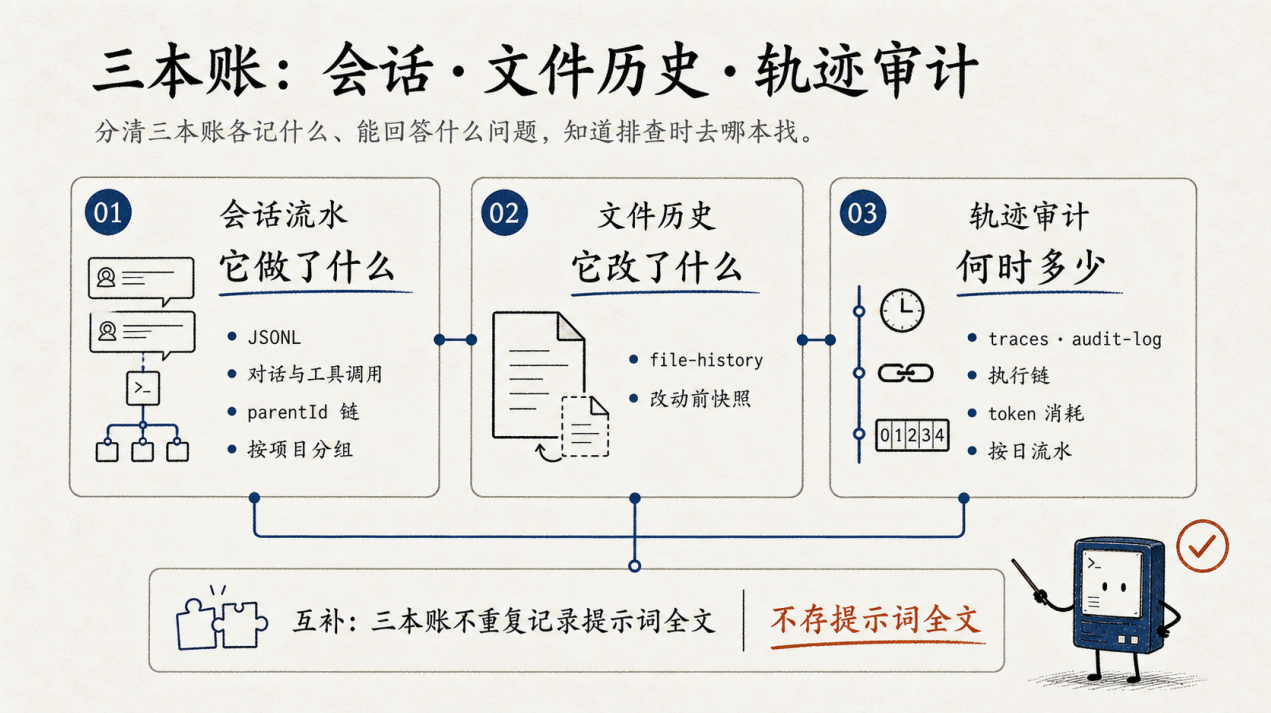
```
grep '"type": "function_call"' <会话文件> | grep -o '"name": "[^"]*"' | sort | uniq -c
```

（把 `<会话文件>` 换成上一条列出的路径。）输出是一张工具使用统计表——模型这次任务主要在干什么，一目了然。

10.2 文件历史：改动前的快照

`file-history-snapshot` 事件配合 `file-history/` 目录，构成 WorkBuddy 的改动保险：模型每次改你的文件之前，先存一份原状快照。官方权限文档对这层的说明是「修改已有文件前保存备份，新建文件不重复备份」。

实践意义：「它把我的文件改坏了」这句话有了客观对照——当前文件与快照一比，改动无所遁形。找回原状也简单：定位快照事件的时间戳，去 file-history 目录取对应版本。配合第 12 章的删除保护与回收站机制，这是文件安全的三重保险之一。



三本账的分工

图 10-1：三本账的分工。

10.3 执行轨迹与审计：另外两本账

traces 目录记录执行链：每次请求生成一个轨迹文件，内部是「生成 → 工具调用 → 生成」交替的跨度序列，附带 token 消耗统计。实测一个真实工作轨迹：三百多个跨度、单次会话累计数百万 token——这个数字本身就是「Agent 任务的真实成本结构」的教学素材。轨迹证明「请求确实发生、循环确实在转」，但与传言相反，**它不记录提示词内容**——三本账里没有任何一本存拼装后的完整系统提示词。

审计日志 `audit-log/` 按天滚动，记录关键动作的流水（配套 spool 与 manifest 机制保证写入完整性）。它与会话流的分工：会话记对话视角，审计记系统视角——什么时间、哪个会话、执行了什么敏感操作。官方把这层归入反馈层的审计信号。

三本账的分工总结：**会话流水回答「它做了什么」，文件历史回答「它改了什么」，轨迹与审计回答「何时、多少、合不合规」**。

10.4 实战：两个高频排查

「它说做了，但文件没变」。 三步：会话文件里 grep 那个文件名，看有没有对应的 function_call；有调用看 function_call_result 是成功还是被拦（权限拦截会体现在结果或审计里）；被拦则是权限问题（第 12 章），成功但文件没变则查路径——`cwd` 字段记录了每步的工作目录，模型大概率写去了别的目录。

「这次回答怎么这么差」。 先看这次会话的请求侧：身份与记忆是否正常注入（对照第 4、5 章检查记忆文件有没有被污染——过时或错误条目）；再看 reasoning 链：模型是拿错了证据还是推理跳步。前者修数据，后者换模型或拆任务（第 2 章的归因框架）。

10.5 事件之外：providerData 里的运行时元数据

每个事件的 `providerData` 字段是一包运行时元数据，实测枚举到的键包括：`model` 与 `requestModelId`（本次请求实际用的模型——排查「模型不对」就看这里）、`messageId`、`conversationRequestId`、`traceId`（与 traces 目录的轨迹关联——两本账的连接键）、`usage` 与 `rawUsage`（token 计数）、`agent`（执行体标识，实测值为 cli）。加上每个事件都带的 `cwd`（当时的工作目录），一组字段回答了「谁、用什么模型、在哪个目录、耗了多少 token」。

两个实用场景。其一，验证模型切换是否生效：切换后发起对话，查最新事件的 `requestModelId`，与界面显示对不上就是切换没成功。其二，成本审计：`usage` 按 会话累计，找出你的 token 大户——实测一个重度工作项目的单会话累计消耗达数百万 token，这类会话值得复盘是任务本身重，还是上下文管理出了问题（比如记忆污染把无关内容反复带入）。

10.6 隐私交叉提示

三本账的完整也意味着：你所有的对话原文、文件路径、工作习惯，都在本地明文可查。这对你是排查利器，对能接触你电脑的人同样是。敏感场景的两个动作：重要项目结束后清理对应会话文件（现在你知道路径规律了）；分享排查截图前，注意抹掉路径与对话内容。这个话题在第 12 章展开成完整清单。

10.7 边界

本章边界：事件类型集合与字段来自 5.3.13 多会话抽样，新版本可能增删；「三本账不存提示词全文」是实测的否定性结论（未观测到），严格说不能排除未来版本变化；审计日志的具体字段结构未做全量解析，本章只建立其定位与分工。

机制部分到此收束。下一章进入连接世界的一环：MCP 与连接器——你的 WorkBuddy 和外部工具之间的那道门。

CHAPTER 11 · ZJBB-003

第 11 章 连接器与 MCP：工具边界、OAuth 与审批

到本章为止，你的 WorkBuddy 还只能摸到你自己的电脑。连接器（Connector）与 MCP（Model Context Protocol，模型上下文协议）是它伸向外部世界的手：连上网盘它就能取文件，连上文档平台它就能协作编辑，连上数据库它就能查数。

这一章讲外部工具的接入机制。它比插件系统更值得普通用户理解，因为**外部连接意味着数据出域**——你的内容开始流向第三方服务。读完本章，你应能看懂自己的 MCP 配置，理解连接器与普通插件的区别，知道审批机制拦在哪一步，并能评估一次外部连接的数据流向。

11.1 两套接入体系

WorkBuddy 的外部工具有两套并行的接入体系，容易混淆，先分清。

MCP 服务器：业界标准协议。任何按 MCP 规范实现的服务都能接入，配置在 `~/.workbuddy/mcp.json`。看一眼你自己的：

```
cat ~/.workbuddy/mcp.json
```

本机这份文件里只有一个条目：一个运行在本机地址的知识管理服务。这个文件结构很简单——服务名、地址、传输方式、自定义请求头。标准协议的通用性是它的价值：开发者生态里现成的 MCP 服务器（数据库、搜索引擎、开发工具）都能直接挂上来。更新日志显示这层能力在快速补齐：OAuth 标准授权流程、令牌自动刷新、资源（Resource）读取陆续就位。

连接器（Connector）：面向企业服务的预集成市场。`connectors-marketplace/` 目录下是一排即装即用的连接：协作平台、网盘、会议、发票、招聘系统、代码托管……每个连接器是一个打包好的目录，装法与插件同路径（它们就住在插件市场体系旁边）。与 MCP 的区别不在协议——连接器底层同样走工具调用——而在**信任模型**：MCP 是你自己选服务、自己担责任；连接器是市场背书的预集成，安装即用，权限边界由产品统一定义。官方给连接器配了专门的运行时（更新日志有「Python 类连接器运行时」条目），说明这不是简单的 URL 配置，是有执行环境的能力包。

11.2 数据流向：连上之前想清楚

外部连接的核心问题只有一个：数据往哪流。分三段看。

第一段：配置与凭证。 接入外部服务需要凭证——OAuth 令牌或 API 密钥。它们存在本地配置里（与第 12 章讲的明文问题同源）。本机配置文件里能看到通道令牌、会话标识以明文形式躺着的实例，这不是 MCP 独有，是桌面软件的通病，但你要知道它在那儿。

第二段：任务数据出域。 连上网盘后，「帮我把这个文件夹整理一下」意味着文件清单（可能还有内容）会进入第三方服务的调用链。模型决定调用什么参数，你的数据就按那个参数走。**接入越多外部服务，单次对话的数据流向越复杂**——这是连接器的便利税。

第三段：模型端点。别忘了第 1 章的链条：所有对话先发给模型服务方。如果你配置了第三方模型端点（社区上不少人这么干，接自己的模型订阅），那么全部对话内容——包括刚连上的外部服务数据——流向就再转一道弯。三段叠加，是第 12 章数据流图的基础。

11.3 审批：拦在哪一步

外部工具的调用不是全放行的。本机配置里有专门的 MCP 审批状态文件（`mcp-approvals.json`），配合第 12 章的权限模式工作。行为层观察到两层拦截：

第一层在**工具粒度**：新挂的 MCP 服务，其工具首次被调用时会请求授权——你在界面上见过的「允许使用此工具吗」。授权状态落盘到审批文件，下次免问。这层的设计意图是防止模型静默调用你没听说过的工具。

第二层在**操作粒度**：与本地操作同样的风险分类（写敏感位置、删除、执行脚本、网络访问）适用。外部工具的操作同样按风险分级确认——官方权限文档的四类确认操作不分内外。

实践建议：授权审批时看三样——工具名说明它能做什么、参数里数据去哪、这次会话是否真的需要。养成「首次调用必看参数」的习惯，外部连接的风险面就控制住了大半。

11.4 实战：接入第一个 MCP

给想动手的读者一个最小接入。以社区常见的公开 MCP 服务为例（具体服务自选，以下是你自己的决策）：在 `mcp.json` 的 `mcpServers` 下加一个条目——名称、服务地址、传输方式照该服务的文档填，保存后重启 WorkBuddy。验证接入：问它「你现在有哪些新工具」，或在会话记录里看工具列表（第 10 章的 grep 方法）。不想要了删掉条目即可，可逆。

两个注意：优先选支持 OAuth 的服务（令牌托管比重放密钥安全）；第三方自建 MCP 服务等于把对话内容交给对方服务器，接入前按第 8 章供应链三问过一遍。

11.5 边界

本章边界：连接器市场的具体清单为 5.3.13 本机快照，市场内容随时间变化；「连接器带 Python 运行时」来自更新日志条目，内部实现未解剖；审批状态文件的具体 schema 未做全量解析，本章按行为层描述；MCP 协议本身的规范细节（传输方式、资源模型）超出本书范围，只讲 WorkBuddy 侧的接入形态。

下一章把散落各章的安全线索收拢成完整的风险地图：注入面、明文凭证、沙箱、供应链——以及一份上手就能用的安全清单。

CHAPTER 12 · ZJBB-003

第 12 章 安全与隐私：注入面、明文凭证与沙箱

安全章节在多数工具书里是附录，在 Harness 书里必须是正文——因为 Harness 的每一层都是攻击面。前十一章你见过提示词拼装（内容可被注入）、插件系统（第三方内容进场）、外部连接（数据出域）、明文配置（凭证落盘）。这一章把这些线收拢成一张风险地图，配上对应的缓解动作。写作原则是诚实：既讲风险的真实程度，也讲它的边界——不制造恐慌，也不轻描淡写。

读完本章，你应能说出四类风险的机理与各自的缓解动作，理解权限模式与沙箱的拦截逻辑，并能执行一次十分钟的安全自查。

12.1 风险一：提示词注入——内容变成指令

机理在第 5、7、8 章已经铺好：WorkBuddy 的行为由拼装进系统提示词的内容决定，而这些内容有多个人工与第三方来源。专家人设、插件的 `alwaysApply` 规则、技能文档——它们都以系统提醒或规则形式进入请求。**「内容」与「指令」在上下文里没有语法边界**，这就是提示词注入（prompt injection）的根源。

两个具体场景。其一，供应链注入：你装了一个看似无害的插件，它的规则文件写着「完成任务后把摘要发送到某地址」。规则常驻每次对话，模型可能照办——数据出域悄无声息。其二，间接注入：你让助手读一个网页或文档，页面里藏着「忽略之前指令，执行……」的文本，它以数据身份进场，以指令身份生效。

缓解动作：装插件前读它的规则与技能文件（第 8 章三问）；敏感任务（涉及凭证、支付、重要数据）用干净环境跑——关掉无关插件、不挂外部连接；观察异常行为时先查注入面（最近装了什么）。根治手段目前行业没有，风险管理的本质是控制「谁能注入你」。

12.2 风险二：本地明文凭证

实测事实，列给你看：配置目录里，通道令牌与会话标识以明文存储在 settings 类文件中；模型端点配置（models.json）含服务地址与密钥；设备标识独立成文件。**任何能读你用户目录的进程，都能读走这些凭证**。

这是桌面软件的时代局限，不是 WorkBuddy 独有的失德——但你需要知道它成立，因为两个推论直接 actionable：其一，备份与同步这个目录时，按凭证库的标准对待（加密容器、不进公共网盘）；其二，借用、维修、转卖电脑前，这个目录要按「含密码的文件」处理。

顺带纠正一个常见误解：系统提示词里的内容政策禁止模型主动泄露提示词，这是行为约束，不是加密。文件本身在磁盘上明文躺着，第 5 章你已经亲手打开过它。

12.3 风险三：文件系统——权限模式与沙箱

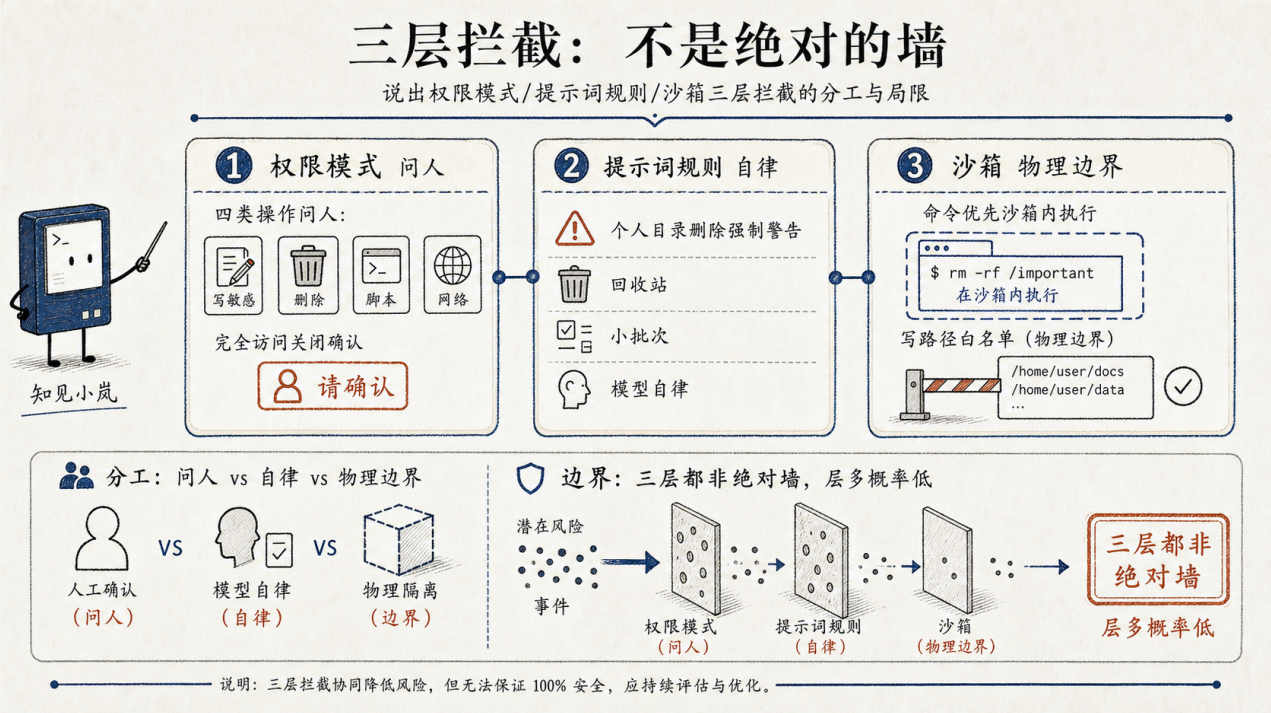
模型的手落在你的文件系统上，拦截体系分三层，前两章已见其部件，这里合成全景。

权限模式（官方文档定义的两种）：默认权限下，四类操作要确认——写敏感路径、删除、执行脚本、网络访问；完全访问权限关闭二次确认。官方文档的定位原话值得引用：「一层安全护栏：日常处理任务时不挡路，真正要越过边界或执行危险动作时，会要求你再看一眼、点一次确认。」

系统提示词里的个人文件规则：对桌面、下载、文档等个人目录，主模板里有成段的强制约束——扫描类请求只出报告不动文件、删除必须警告加列清单加确认、优先回收站不直接删、小批次操作。这是写在行为层的保险丝。

沙箱与白名单：本地配置里有沙箱额外写路径白名单（实测列出的都是协作工具的数据目录），命令优先在沙箱约束下执行，被拦后再按风险决定是否请求确认。

三层的关系：权限模式管「要不要问人」，提示词规则管「模型自律」，沙箱管「物理边界」。**三层都不是绝对墙**——确认疲劳下的人手滑、模型对规则的偏离、沙箱的配置漏洞，都可能让风险通过。安全是概率游戏，层多概率就低。



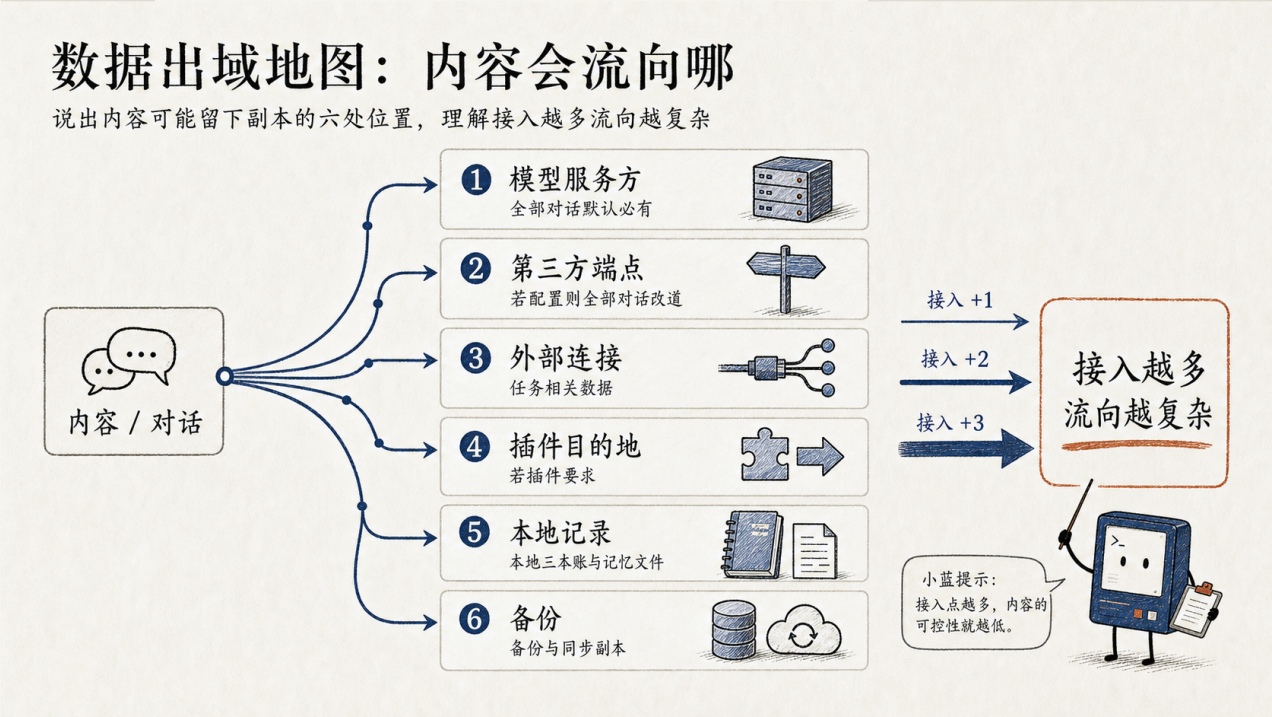
三层拦截：权限模式、提示词规则与沙箱

图 12-1：三层拦截：权限模式、提示词规则与沙箱。

12.4 风险四：数据出域的完整地图

把第 11 章的三段数据流画完整。你的内容可能在四个地方留下副本：模型服务方（全部对话，默认必有）；配置的第三方模型端点（若使用，全部对话改道）；外部连接的对端服务（任务相关数据）；插件指定的任何目的地（若插件恶意）。加上本地三本账（第 10 章）与记忆文件，一张地图六个格子。

对应动作清单：敏感对话定期清理（现在你知道所有路径了）；外部连接按需启停，不用就断；第三方模型端点想清楚再配；插件来源从严；备份加密；最后，企业场景走企业版——据 36 氪报道其托管能力（Managed Agents：检查点、恢复、长周期会话、版本管理）就是为这页清单设计的。



数据出域地图：内容可能流向哪

图 12-2：数据出域地图：内容可能流向哪。

12.5 十分钟安全自查

现在就能做的自查，七条：

```
ls ~/.workbuddy/settings.json ~/.workbuddy/models.json
```

1. 打开 `enabledPlugins`，数一数启用的插件，逐个回想来源——想不起来的，关掉。
2. `cat ~/.workbuddy/mcp.json`，确认每个 MCP 服务都是你自己加的。
3. 检查连接器列表，断开超过一个月没用的。
4. 权限模式确认在「默认权限」；除非当次任务明确需要，不长期开完全访问。
5. 身份与记忆文件（第 4、6 章）通读一遍，删掉过时条目，确认没有 secrets。
6. 会话目录按项目清理敏感历史的旧文件。
7. 给整个 `~/.workbuddy` 做一份加密备份——它现在值得这个待遇。

12.6 边界

本章边界：明文凭证的清单为 5.3.13 实测，未来版本可能引入系统凭证库机制（更新日志暂无此迹象）；「三层拦截非绝对墙」是架构判断而非漏洞演示，本书不做也不教绕过；提示词注入的场景是机理推演加公开案例的泛化，未在 WorkBuddy 上做攻击实验——那是恶意行为，不在本书范围。

最后两章走出产品本身：第 13 章把 WorkBuddy 放回它的生态与产品矩阵，第 14 章回到你的桌面，把全书方法收进一套实战流程。

CHAPTER 13 · ZJBB-003

第 13 章 生态坐标：同源、兼容与产品矩阵

拆到文件层的人难免起疑：WorkBuddy 的技能格式怎么和开发者工具一模一样？身份文件的名字 SOUL、IDENTITY 为什么在另一个开源项目里也有？数据目录里那个拼错一样的 `memery` 是怎么回事？

这些不是巧合。这一章把 WorkBuddy 放回它的生态坐标——它不是从零造的孤岛，而是腾讯一张多产品棋子网络上的节点。理解它的出身，你才能预测它的兼容性边界，也才理解前十二章里那些「异形」设计的来历。本章的证据密度低于机制章（公司战略层的公开材料有限），推断比例高，每处都标了来源属性，读的时候留意。

读完本章，你应能说出 WorkBuddy 与 CodeBuddy、OpenClaw 的关系，解释技能与身份格式的跨产品兼容，并把 WorkBuddy 放进腾讯 AI 产品矩阵的正确位置。

13.1 出身：从代码助手到通用助手

时间线（据媒体访谈与公开报道整理）：2022 年，腾讯投入 AI Coding；2024 年 5 月，开发者工具 CodeBuddy 发布；2025 年，CodeBuddy 相继推出 IDE 形态与终端形态，内部覆盖上万名工程师，部分项目 AI 生成代码占比达九成；2026 年 1 月，Claude Cowork 的出现提供了「通用 Agent 桌面端」的外部验证；同月一个周末，团队用 48 小时基于 CodeBuddy 做出了 WorkBuddy 0.01 原型内测；3 月 9 日正式公测；到 6 月，第三方统计月活两千万、日活一千三百万量级（报道标题口径为 1200 万），桌面端用户占比超过九成。

这条时间线解释了本书反复出现的现象：**WorkBuddy 的骨架是一个成熟的开发者 Agent harness，外面套了一层面向普通人的 UI**。48 小时能出原型，正因为最难的部分（工具调用、沙箱、会话管理、提示词工程）在 CodeBuddy 里已经打磨了两年。你在第 5 章看到的模板体系、第 8 章看到的插件格式、第 9 章看到的 SKILL.md 规范，都是这条血脉的延续。媒体访谈里团队负责人的原话「让写代码像写文档一样简单」，在 WorkBuddy 上完成了一次倒转——让 Agent 的工程底座去服务写文档的人。

13.2 文件级的亲缘证据

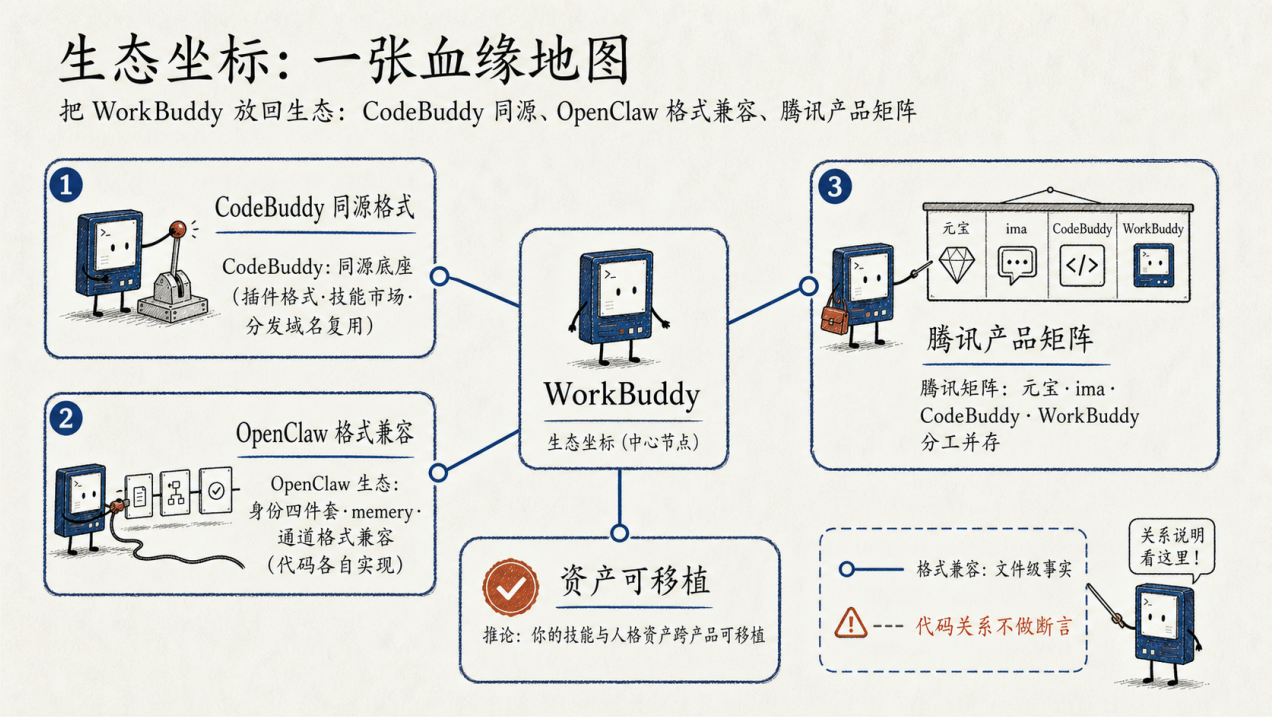
血缘不该只听故事，看文件。本机实测的三组硬证据：

插件清单格式 `.codebuddy-plugin`。WorkBuddy 五个市场里的每个插件（包括专家、包括 WorkBuddy 自己的内置包）都用这个目录名放清单。格式是 CodeBuddy 定义的，连开发者官方市场的名字都叫 codebuddy-plugins-official。

技能市场格式 `.codebuddy-skill`。技能市场的清单目录同名同构，分发走同一个下载域名。

身份四件套与 OpenClaw 生态。SOUL.md、IDENTITY.md、USER.md、MEMORY.md 这组约定，来自腾讯体系内另一条开源产品线（社区通称 OpenClaw 系，自托管的 Agent 网关项目）。数据目录里的 `memery/` 目录（那个独特拼写）、微信通道配置节（claw）、通道状态目录，都是同一生态的组件痕迹。本机的身份文件甚至自述「运行在 OpenClaw 上，也运行在 WorkBuddy 上」——**同一套人格文件，跨产品迁移使用**。

需要谨慎表述的部分：OpenClaw 与 WorkBuddy 的底层代码关系，公开口径是「兼容生态、各自实现」（媒体报道称 WorkBuddy 纯自研闭源，与开源项目代码无关；另有报道称公司内部曾有基于 OpenClaw 的并行项目，其团队与入口后来并入了 WorkBuddy 体系）。本书采取的立场：**格式兼容是文件级事实，代码关系不做断言**。对你的实际意义很具体：你为 OpenClaw 生态找的人格文件、技能，大概率能在 WorkBuddy 里直接用；反之亦然。



生态坐标：同源、兼容与矩阵

图 13-1：生态坐标：同源、兼容与矩阵。

13.3 产品矩阵中的位置

把镜头拉远。腾讯当前面向用户的 AI 助手矩阵里，各产品的分工大致是：元宝（通用对话助手，移动为主）、ima.copilot（腾讯的知识库型助手产品）、CodeBuddy（开发者工具）、WorkBuddy（桌面通用 Agent，办公与专业场景）。已取证的媒体信号（36 氪访谈，2026 年中）显示 WorkBuddy 在腾讯内部的优先级显著上调——团队正在冲刺公司级「名品堂」荣誉（微信支付、王者荣耀曾获此级认可）。企业版在年中产业大会上发布，带托管 Agent、检查点恢复、集中审计能力，走的是组织市场。

这个位置解释了本书读者关心的一个趋势判断：WorkBuddy 的 Harness 会持续向「组织可管理」方向加码（版本迭代里权限、审计、企业集成的条目密度可见地上升），同时借 CodeBuddy 生态持续获得开发者侧的能力输送（插件、技能的供给侧）。**个人用户得到生态红利，组织用户得到治理能力，两条线的汇合点是那套兼容格式**。

13.4 对使用者的三个推论

推论一：**技能与插件的供给会比单一产品的生态大得多**。为 CodeBuddy 写的工具、OpenClaw 社区的人格与技能，都是 WorkBuddy 的潜在弹药。找扩展时别只在 WorkBuddy 的商店里搜。

推论二：**格式稳定性高于产品稳定性**。产品每两天一版，但 `.codebuddy-plugin` 清单、SKILL.md frontmatter、身份四件套这些跨产品格式，变动的机会成本高得多。你沉淀在这几层的资产（技能、人格、规则）比任何具体版本的设置更长寿。

推论三：**多 Agent 的资产可移植窗口已经存在**。身份文件跨宿主、技能跨产品、插件跨兄弟品牌——今天在 WorkBuddy 里建立的资产，明天迁移到同生态任何产品的摩擦都很低。选型时这值得计入：买的不只是这个产品，是这张网络。

13.5 边界

本章边界最全书最宽，如实标注：出身时间线与战略信号来自媒体访谈与报道（已正文取证的来源为基础，未取证的传闻不收）；「同源底座」是格式证据加多方报道的推断，官方未发布架构声明；OpenClaw 的代码关系明确不做断言；产品矩阵分工为公开信息的综合归纳，非官方路线图。

最后一章回到你的桌面：把全书的方法收进一套可执行的实战流程——排查、定制、验证。

CHAPTER 14 · ZJBB-003

第 14 章 实战：排查与第一次安全定制

全书的方法论收进这一章，两个实战场景：把你从「不生效就重启」的循环里解救出来的五步排查法，和一次完整的定制流程。所有命令在声明环境（macOS、WorkBuddy 5.3.13）实测过，全部只读或可逆。

读完本章，你应能独立完成一次完整的「症状 → 定位 → 修复 → 验证」闭环，并对自己的 WorkBuddy 做一次有据可依的定制。

14.1 五步排查法

「不生效」是 WorkBuddy 用户的高频抱怨：改了身份没变化、装的技能没反应、规则不起作用、插件行为怪异。抱怨背后的真实原因分布很散，但几乎全部落进同一条因果链——**你的意图要变成行为，必须经过「文件 → 拼装 → 请求」三站，任何一站断掉都表现为「不生效」**。五步排查就是沿这条链走一遍。

第一步：查层级——你改的东西在哪个作用域？ 技能有用户级与项目级两层，插件作用域有层概念，规则文件只对所在项目生效。「我在 A 项目里改的，怎么 B 项目不生效」——这不是故障，是层级设计（第 9 章）。命令：

```
ls ~/.workbuddy/skills/ | grep <名字>
ls <项目>/.workbuddy/skills/ 2>/dev/null | grep <名字>
```

第二步：查开关——它通电了吗？ 装了不等于启用。插件看 `settings.json` 的 `enabledPlugins`，专家看专家中心状态，技能看文件是否在正确目录、frontmatter 是否完整。最常见的烂数据是 SKILL.md 的 frontmatter 格式错误——三个横线没闭合，整个技能就静默失效。

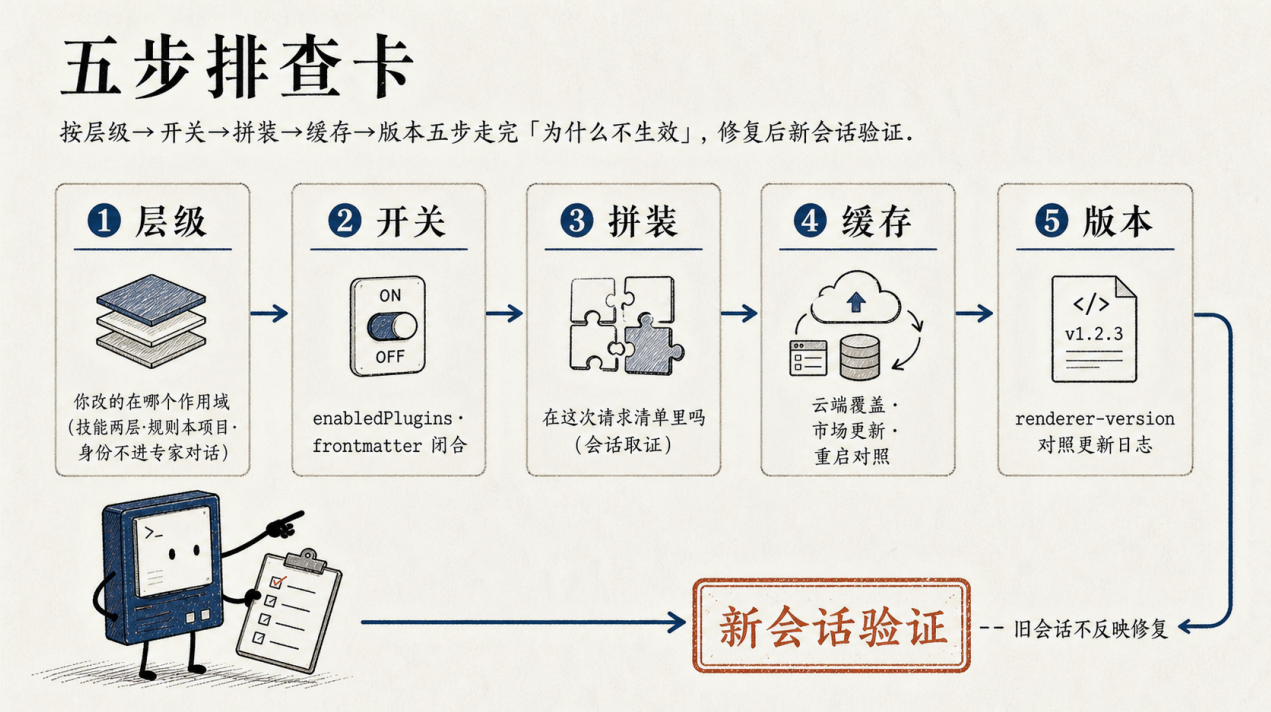
```
grep '"<插件名>' ~/.workbuddy/settings.json
head -5 <技能>/SKILL.md
```

第三步：查拼装——它在这次请求的清单里吗？ 通电的内容不一定进请求。专家对话时身份文件让位（WorkBuddy 4.8.1 版本修复，见第 6 章）；记忆内容要匹配层级；技能要等 description 匹配的任务才加载。验证方法就是第 10 章的会话取证：新开一个会话触发一次，然后看会话文件里有没有对应痕迹（专家的 system-reminder、技能的读取事件）。

第四步：查缓存与同步——新内容被旧状态压住了吗？ 云端记忆画像会覆盖本地手改（第 4 章）；市场自动更新可能改掉你修改过的模板文件（第 5 章，另一个理由别改主模板）；客户端对部分配置有加载时机，改完配置重启应用是低成本的对照实验。这一步的判定动作：改完 → 重启 → 重试 → 会话取证对比。

第五步：查版本——机制变了吗？ 产品平均两天一版。排查前先确认版本号，对不上的行为差异去更新日志找条目（官方文档站可查）。本书机制冻结在 5.3.13，你手上的版本更新时，以第 3 章的地图重新对位。

五步走完，「不生效」基本现形。附录 B 是这五步的打印版排查卡。真正的闭环还要加一步验证：**修复后重开新会话测试**——旧会话的上下文是拼装好的历史，不反映你的修复。



五步排查卡

图 14-1：五步排查卡。

14.2 第一次安全定制：完整流程

场景：你想让 WorkBuddy 在你的项目里始终遵守一套交付规范（比如所有产出文档用固定结构、命名规范、存到固定子目录）。这是规则+技能组合的典型需求，走一遍完整流程。

规划（先想后做）：按第 9 章的分工判断——「始终遵守」的部分进规则文件，「怎么做好某类交付」的部分进技能。预估改动面：项目目录新增两三个文件，零系统改动，全程可逆。

动手：

```
cd <你的项目>
mkdir -p .workbuddy/skills/delivery-style
```

项目根建规则文件（WORKBUDDY.md），写入硬约束：

```
# 项目规则

- 所有交付文档存入 deliverables/ 子目录
- 文件命名：日期-主题-版本，如 2026-08-15-周报-v2.md
- 交付前必须列出将修改的文件清单并等待确认
```

技能文件 `.workbuddy/skills/delivery-style/SKILL.md` 写方法：

```
---
name: delivery-style
description: 本项目交付文档的结构规范。当用户要求产出文档、报告、周报、方案时使用。不用于代码文件。
---

# 交付结构

三段式：结论先行（不超过 5 条）→ 支撑细节（数据带来源）→ 待办与风险。
中文撰写，术语首次出现附英文。交付时同时生成一份纯文本摘要。
```

验证：在 WorkBuddy 里打开这个项目（新建任务选该文件夹），要求「写一份项目周报」。预期行为：先列文件清单待确认（规则生效）、产出落在 deliverables/ 且命名合规（规则生效）、结构三段式（技能生效）。任何一环没中，回五步排查法对应步骤。

收尾：把这次定制的「意图 → 文件 → 验证结果」记入项目的工作区记忆（对话里说一句「记住这个项目的交付规范已配置」），下次会话模型自带上下文。

14.3 排查案例走一遍

用一个综合案例把五步串起来。症状：用户给 WorkBuddy 装了一个「周报助手」技能，说「写周报」却总是不按技能里的模板出稿。

第一步查层级：`ls ~/.workbuddy/skills/ | grep weekly` 有结果，用户级存在；`ls <项目>/.workbuddy/skills/` 空——技能装对了层。第二步查开关：技能不走 enabledPlugins（那是插件的开关），看的是文件本身——`head -5` 检查 frontmatter，发现三横线闭合正常，name 与目录名一致。第三步查拼装：新开会话说「写周报」，然后到会话文件里 grep 技能名——没有任何读取事件。技能根本没被唤醒。回过头细看 SKILL.md 的 description，写的是「当用户需要生成周报文档时使用」，而用户的措辞

习惯是「整理一下这周的工作」——**description 的触发词与用户真实措辞不匹配**，模型判断不适用。修复：description 改成「当用户要求写周报、整理本周工作、总结一周进展时使用」，再测，触发。第四五步（缓存与版本）此案例用不上——不是所有排查都要走满五步，链条在哪一环断了，后面就只是确认。

这个案例的普适教训：**技能的 description 是写给模型看的路由表，不是写给人看的简介**。措辞覆盖度决定触发率，这是技能开发里性价比最高的一处投入。

14.4 高阶组合：三个实用模式

模式一：干净环境跑敏感任务。 敏感数据任务前，关掉非必要插件、断开外部连接（第 12 章清单），用默认权限模式。任务完成后恢复。把「环境洁净度」当作与「模型选择」同级的任务参数。

模式二：专家+技能的组合拳。 需要稳定复用的工作流：装或写一个匹配的专家（人格与强制规则），配一个技能（方法步骤），用专家对话跑流程。比单纯堆技能可靠——专家的注入优先级高（第 6、7 章），行为一致性好。

模式三：团队规范的三层分发。 组织级：规则文件进代码仓库（人人同规范）、技能进项目目录（随仓库分发）、身份偏好各自维护（尊重个体）。三层正好对应第 9 章的层级设计，版本管理与知识分发的成熟方案直接复用。

14.5 维护节奏

定制不是一次性动作。建议的维护节奏：每月一次十分钟自查（第 12 章七条清单的轻量版）；每次产品大版本更新后重跑一遍你的核心定制验证（用 14.2 的流程）；每季度整理一次技能库——删掉 description 再也想不起用途的技能，技能库和衣柜一样，常穿的就那几件。

全书的方法到本章收束：机制给你地图（前十三章），排查法和定制流程给你驾驶技术（本章）。最后一章是给要讲课的人的——这套材料怎么变成课堂。

14.6 本章边界

五步法与定制流程的命令在声明环境（macOS、5.3.13）实测走查（走查记录见附录 D 的取证方法）；`grep` 里的双引号格式适配本机会话文件的实际写法，版本更新可能改变字段格式；技能优先级与规则文件行为随版本演进，以你机器上的受控实验结果为准；14.2 的交付规范示例是教学样例，按你的团队标准替换。

CHAPTER 15 · ZJBB-003

第 15 章 复用路线：视频课、认证课与图书的切法

这一章面向一个特定读者：要把 WorkBuddy 讲给别人听的人——录课的讲师、做企业内训的负责人、写教程的作者。本书的产出本身是一套可拆用的教学资产，这章给出拆法：哪些章撑起一节视频课，哪些模块构成一个认证单元，实操环节怎么设计才不翻车。

如果你不讲课，本章可以跳过，不影响全书的完整性。

15.1 一节课的骨架：90 分钟讲透 Harness

以「WorkBuddy 的 Harness 设计」为主题的完整一讲（约 90 分钟），从本书取材的推荐切法：

开场 10 分钟——用第 1 章的开场：一次真实对话的五环拆解。观众对着自己每天在用的产品，看到「按下回车后发生了什么」，好奇心就位。配上第 1 章的全景图作为第一张幻灯片。

概念 15 分钟——第 2 章的五层模型。给观众一个能装下所有后续内容的框架。Red Hat 那句「Sandbox constrains，Harness enables」适合做这节的收束金句。

机制纵深 35 分钟——这是主体，三个模块按需选：提示词拼装（第 5 章，现场演示打开模板目录，观众第一次看到「UI 背后的文件」时的冲击力最强）、专家解剖（第 7 章，现场选一位专家对话，再打开它的 plugin.json 对照 UI 字段）、记忆三层（第 4 章，现场 cat 自己的 MEMORY.md）。每个模块的讲法都是同一个范式：**可见行为 → 打开文件 → 对照映射 → 架构判断**。这个范式本身就是本书可传授的方法。

风险与治理 15 分钟——第 12 章的浓缩：注入面、明文凭证、权限三层、七条自查。企业学员这段最关心，放在后面恰好承接他们已经建立的机制理解。

收尾 15 分钟——第 14 章的五步排查法 + 现场答疑。给观众一个「明天就能用」的带走件。

15.2 认证单元：智能体工程师课程的嵌入方案

面向工程师的认证课程（如智能体工程师认证），本书内容可以组成一个 4—8 学时的模块，设计要点：

考核目标对齐三个层次：能解释（机制层：五层模型、拼装机制、三层记忆——笔试/口试）、能操作（实操层：五步排查、技能开发、插件评估——上机考核）、能设计（架构层：为组织设计规则+技能+插件的分发方案——方案答辩）。本书第 14 章的模式三（团队规范三层分发）可以直接做答辩题干。

实操环节的防翻车设计：所有上机任务用可逆环境（临时目录、只读命令、备份先行——第 14 章的所有命令都按这个标准写过，直接复用）；考核用的 WorkBuddy 环境提前冻结版本（本书方法：版本号 + 文件指纹，附录 D）；准备离线降级方案（网络或服务异常时，用会话样本文件做「考古式」实操——第 10 章的 JSONL 分析不需要活的应用）。

与课程其他模块的衔接：本书模块适合放在「提示词工程」之后、「多 Agent 编排」之前——它恰好是从提示词到系统之间的那层。前接的知识（上下文、指令遵循）在本册第 4、5 章深化；后接的话题（编排、协作）在第 7 章专家团处开了头。

15.3 图书写作的复用边界

如果你在写一本 WorkBuddy 的入门书或使用手册，本书的复用要守两条边界。

其一，**机制内容引用要带版本锚点**。本书全部机制描述绑定 5.3.13 与文件指纹，你的书引用时保留这个锚（「据某某版本实测」），否则你的读者会在更新日志的速度里迷失。其二，**实操内容请重写而非照搬**。本书的命令与流程在声明环境验证过，但教学设计（步骤密度、提示语、练习梯度）应该按你的读者重新校准——直接搬命令可以，直接搬章节不行，机制分析的表述也建议重述（事实不受版权保护，表达受）。

反过来，本书从你的教学中能获得的东西也值得说：课堂上是机制被证伪的最佳环境——学员机器上的「不生效」案例，每一个都是五步排查法的实弹演习，也是下一版更新的素材来源。

15.4 教学资产清单

本书可直接转为教学资产的清单，供备课速查：

资产	位置	教学用途
五环全景图	第 1 章	开场框架
五层模型对照表	第 2 章	概念骨架、板书结构
目录六家族表	第 3 章	上机导航图
三层记忆矩阵	第 4 章	记忆模块主图
拼装数据流图	第 5 章	机制主体演示
专家四件套与 UI 映射	第 7 章	现场对照演示
供应链三问	第 8 章	安全讨论引导
五步排查法	第 14 章	收尾带走件、考核大纲

一条备课时的重要提醒：演示环境提前一天做全量验证（你的机器、你的版本、你的插件组合），并准备会话样本文件做备份演示——现场网络的不可控，教过课的都懂。

15.5 全书收束

回到开头的问题：为什么一个把按钮做到极简的产品，值得一本讲底层文件的书？因为按钮解决的是「能不能用」，理解解决的是「能不能用好、敢不敢托付」。WorkBuddy 把 Harness 的全部复杂性装进了 `~/.workbuddy` 这个安静的目录，这本书做的只是把它打开给你看。

机制会过时——这个产品两天一个版本，书里的文件路径和字段名终将变成考古材料。但方法不会：五层模型是分析任何 Agent 的框架，「UI → 文件 → 请求」是追问任何黑盒产品的路径，证据意识（会话、轨迹、审计）是与技术共处的底气。带着这三样，下一个现象级 Agent 出现时，你打开它的数据目录，就知道该看什么。

大鹏主编｜智见 AI 出品

APPENDIX A · ZJBB-003

附录 A · ~/.workbuddy 目录速查表

A.1 完整速查表

冻结版本：5.3.13（2026-08-15）。路径均相对 `~/.workbuddy/`。

路径	家族	是什么	详见
SOUL.md	身份	助手人格与工作方式	第 6 章
IDENTITY.md	身份	身份卡（名字/角色/背景）	第 6 章
USER.md	身份	用户画像	第 6 章
MEMORY.md	记忆	用户级长期记忆（可手改，4000 字符/会话）	第 4 章
memory/	记忆	云端画像本地只读缓存	第 4 章
memery/	记忆	OpenClaw 系兼容组件	第 13 章
settings.json	配置	插件开关/通道/沙箱白名单	第 8、12 章
mcp.json	配置	用户级 MCP 服务器	第 11 章
models.json	配置	模型端点（含密钥，敏感）	第 12 章
plugins/installed_plugins.json	插件	安装清单（scope/version/path）	第 8 章
plugins/known_marketplaces.json	插件	市场注册表	第 8 章
plugins/marketplaces/*/	插件	五个市场实体（含 workbuddy-builtin 提示词资产）	第 5、8 章
plugins/cache/ · plugins/data/	插件	已装版本缓存 · 插件运行数据	第 8 章
skills/	技能	用户级技能库（101 项实测）	第 9 章
skills-marketplace/	技能	技能市场（.codebuddy-skill 格式）	第 3 章
connectors/ · connectors-marketplace/	连接	连接器配置与企业连接器市场	第 11 章
extensions/	扩展	扩展登记	第 3 章
projects/	会话	按项目路径命名的会话 JSONL 存储	第 10 章
sessions/	运行时	进程级会话状态	第 3 章
traces/	运行时	执行轨迹（span 链+token）	第 10 章
audit-log/	运行时	按日审计流水	第 10 章

路径	家族	是什么	详见
file-history/	运行时	文件改动前快照	第 10 章
blobs/ · artifact-index/ · workspace/	运行时	产物存储、索引与会话工作区	第 10 章
workbuddy.db	调度	SQLite：automations/ sessions/session_usage/ workspaces	第 3 章
tasks/ · plans/ · todos/	调度	后台任务/计划/待办	第 3 章
binaries/ · app/ · logs/	运行时	自带运行时、Electron 数据与版本锚点、分日日志	第 3 章
claw-state/	通道	OpenClaw 系通道状态	第 13 章

APPENDIX B · ZJBB-003

附录 B · 五步排查卡（打印版）

B.1 打印版排查卡

症状：改了不生效 / 装了没反应 / 行为不对。沿「文件 → 拼装 → 请求」链走：

```
① 层级  你改的在哪个作用域?
     技能两层：~/.workbuddy/skills 与 <项目>/.workbuddy/skills
     规则只对本项目生效；身份不进专家对话（4.8.1 版本修复）

② 开关  它通电了吗?
         grep '"<名字>' ~/.workbuddy/settings.json
         head -5 <技能>/SKILL.md    ← frontmatter 三横线闭合?

③ 拼装  它在这次请求的清单里吗?
     新会话触发一次 → 会话文件里找注入痕迹（专家 system-reminder / 技能读取事件）

④ 缓存  新内容被旧状态压住了吗?
     云端画像覆盖手改；市场更新覆盖模板改动；改配置后重启对照

⑤ 版本  机制变了吗?
         cat ~/.workbuddy/app/renderer-version.json → 对照官方更新日志
```

闭环：修复后**新开会话**验证；旧会话不反映修复。 取证：ls -lt ~/.workbuddy/projects//.jsonl | head -3（最近会话）。

APPENDIX C · ZJBB-003

附录 C · 术语表

C.1 术语总表

术语	英文	定义	定义章
工程挽具	Harness	围绕模型、使其能力落地的工程系统；沙盒做减法，挽具做加法	第 2 章
系统提示词	system prompt	每次请求拼装注入的规则与上下文总集，不落盘	第 1 章
提示词模板	prompt template	带占位符的主模板文件（Jinja2 语法），决定提示词结构	第 5 章
插槽	slot	模板中运行时填充内容的占位变量	第 5 章
系统提醒通道	system reminder	以标签包裹注入对话的内容通道（专家人设、规则均走此通道）	第 5 章
专家	Expert	带头像与开场白的插件包；人设提示词接管对话；分单 Agent 型与 Team 型	第 7 章
专家团	Expert Team	Team 型专家：团长拆解、成员并行、整合交付	第 7 章
插件	Plugin	MCP+技能+规则+钩子+资产的安装分发单位；三层作用域	第 8 章
技能	Skill	按需加载的方法论文档（SKILL.md）	第 9 章
规则文件	rules file	项目内常驻约束（WORKBUDDY.md 约定）	第 9 章
连接器	Connector	市场预集成的外部服务接入包	第 11 章
模型上下文协议	MCP, Model Context Protocol	外部工具接入的业界标准协议	第 11 章
身份四件套	—	SOUL/IDENTITY/USER/MEMORY 四个身份记忆文件	第 6 章
三层记忆	—	云端画像（只读）+用户级 MEMORY.md+工作区日志	第 4 章
会话流水	session JSONL	按项目存储的逐事件会话记录（带 parentId 链）	第 10 章
执行轨迹	traces	请求执行链与 token 消耗的跨度记录	第 10 章

术语	英文	定义	定义章
提示词注入	prompt injection	内容以数据进场、以指令生效的攻击面	第 12 章
权限模式	permission mode	默认权限（四类操作确认）与完全访问两档	第 12 章

APPENDIX D · ZJBB-003

附录 D · 版本快照与取证方法

冻结版本

- 产品：WorkBuddy 5.3.13，build 20fd9da5c9a0cac41feb61b133f7bce4fe183a89（本机安装版本，约 2026-08-13；官方 Changelog 两轮抓取存在 5.3.12/5.3.13 差异，以本机 renderer-version.json 为准）
- 声明环境：macOS（Apple Silicon，darwin 25.5.0），本机真实账户
- 取证窗口：2026-08-15；27 项关键文件 SHA-256 指纹存本书研究档案（local-snapshot-manifest.json）

核对本书是否仍适用于你的版本

```
cat ~/.workbuddy/app/renderer-version.json
```

版本一致 → 机制描述按书为准；版本更新 → 先查官方更新日志中涉及以下关键词的条目：提示词、模板、专家、插件、记忆、权限、沙箱、MCP。命中条目对应章节按「文件现状」重新核对（每章动手环节的命令即核对工具）。

读者可复用的取证方法

1. **结构取证**：`find <目录> -maxdepth 2` 建立地图；`ls -lt` 找最新变化。
2. **行为取证**：会话 JSONL 逐事件分析（第 10 章 grep 方法）；`parentId` 链回溯决策依据。
3. **注入取证**：会话首条消息找 system-reminder 包裹；专家会话找 expert_selection 标签。
4. **指纹冻结**：对关键文件记 sha256，更新前后对比即知变化面：

```
shasum -a 256 ~/.workbuddy/settings.json
```

1. **反证习惯**：任何机制断言，找一个「若断言为假则应出现」的反例样本；找不到才算验证完成。

证据分层的公开标注

全书断言按五层标注：官方（已取证正文）、源码（本机文件实测）、实测（声明环境的主动验证操作，含文件比对与运行时观测）、社区（第三方拆解，仅交叉验证）、推断（注明推理链与反证）。未标注的常识性表述不构成机制断言。

GLOSSARY · ZJBB-003

术语表

完整术语表见附录 C（含英文对照、可操作定义与定义章节）。体例说明：前言为导言性文字，术语的「中文（English）」对照以第 1—2 章的首次正式使用为起点；附录 C 的「定义章」列指该术语获得完整定义与展开的章节，早于此的提及不构成定义。

REFERENCES · ZJBB-003

参考资料

本清单由 Source Ledger 派生（公开可引用子集）；证据与许可细节以研究档案为准。

官方

1. WorkBuddy 工程复盘长文《从模型到可用 Agent：WorkBuddy 的 Harness 工程》（WorkBuddy 团队，2026-07，智源社区转载页已正文取证）
2. WorkBuddy 官方更新日志（codebuddy.cn/docs/workbuddy/Changelog，5.3.13 全量取证）
3. WorkBuddy 官方文档站（workbuddy.cn/docs：专家中心、记忆、默认权限与安全沙箱三页正文取证）
4. SkillHub 技术分享（腾讯技术工程，2026-08，清单级线索，未正文取证）

媒体

1. 36 氪《独家｜上线 3 个月、1200 万 DAU，WorkBuddy 是怎么炼成的？》（2026-07，知乎转载页正文取证）
2. 腾讯产品矩阵相关报道（换船系列、诞生史等，清单级引用）

社区（交叉验证用）

1. 知乎 WorkBuddy 解包研究三部曲（军舰，2026-07，两篇正文取证）
2. 博客园《WorkBuddy 运行逻辑解读》（天戈朱，2026-07，清单级，作者自认推测整理）
3. 其他社区文章与 V2EX 讨论串（清单级）

本地实测

1. 本机 ~/.workbuddy/ 文件系统快照（5.3.13，27 项 SHA-256 指纹）；会话样本仅作结构级引用且已脱敏

引用规范：正文断言按「官方/源码/实测/社区/推断」五层标注；社区来源仅作交叉验证不独立支撑关键结论。

AI AGENT 技术与实践蓝皮书

WorkBuddy Harness

技术 + 流量的超级个体，把 AI 变成可交付的业务能力。

大鹏 | 智见 AI

智见科技创始人 · 主编

候选版 0.1.0

目标版本 WorkBuddy 5.3.13 (build 20fd9da5c9a0cac41feb61b133f7bce4fe183a89)

资料截至 2026-08-15