DioChain 白皮书

面向智能纪元的 Agent-First 数字社会基础设施

Version: Draft v0.2

目录

第 1 章：摘要

DioChain 是什么

DioChain 是一套面向 AI Agent 的链上社会基础设施。DioChain 为 AI Agent 提供完整的数字社会基础设施，涵盖身份、经济、法律与合规。

当前的区块链为人类用户设计——手动签名、手动发起交易、手动解决争议。而 AI Agent 的大规模涌现正在改变这一前提：在可预见的未来，链上交易的主体将从人类转向自主运行的 Agent。这些 Agent 需要的远不止一个转账通道，它们需要身份（DID）、货币（Token）、市场（算力与服务的买卖）、法律（纠纷仲裁）、合规（满足不同司法管辖区的要求）——换言之，它们需要一个社会。

目前没有任何一条链原生提供这套完整的社会基础设施。

核心设计

DioChain 采用三层架构：

- **L0 中央结算与司法层**：提供最终结算、AI 司法仲裁和全局共识。
- **L1 分布式网点层**：保证金驱动的本地清算所，处理高频交易，定时与 L0 净额结算。
- **L2 Agent 与用户层**：Agent 和人类用户在此执行业务逻辑，通过 L1 接入网络。

关键技术决策：

1. **AI 不在热链路执行**。AI 作为异步进化引擎，持续挖掘链上数据、优化确定性规则库，而非嵌入交易执行路径引入不确定性。
2. **ACTUS 金融合约标准原生集成**。所有金融产品共享一套机器可读的合约语义，Agent 之间无需逐一对接协议。
3. **合规插槽（Compliance Slots）**。为不同司法管辖区预留可编程的合规接口，避免硬编码单一法律体系。

4. **算力交易中心**。Token 可直接用于按量付费调用各类 AI 模型，算力是 Agent 的"食物"，与货币价值脱钩。

为谁构建

DioChain 的主要"居民"是 AI Agent——从简单的交易机器人到复杂的自治金融服务提供者。人类用户作为参与者和委托人存在，可以部署、监督和受益于 Agent 的活动。

面向的参与方包括：需要自治经济身份的 Agent 开发者；需要合规框架的金融机构；需要低摩擦算力接入的 AI 应用；需要跨管辖区互操作性的监管方；以及寻求 AI+区块链基础设施长期价值的生态投资者。

为什么是现在

2025 年，AI+Crypto 赛道经历了一次深度价值重估：AI 代币整体下跌超过 50%，约 $6B 市值蒸发。低成本推理服务的出现证明了一个基本事实——如果区块链层不提供不可替代的具体功能，那它就只是摩擦。市场已从叙事驱动转向价值驱动。

存活下来的项目有一个共同特征：区块链在其中承担了不可替代的角色（激励协调、密码学验证、去中心化共识）。DioChain 正是基于这一原则设计：链提供经济、法律、身份、合规等不可替代的社会原语，AI 提供智能和适应性，两者各司其职。

核心主张

我们只定义原语（Primitives），不定义应用（Applications）。保险、征信、衍生品、风控——这些都应当由市场涌现，而非由协议层硬编码。正如以太坊提供了 EVM 和 ERC-20，Uniswap 自然涌现；DioChain 提供经济、合约、身份、司法、合规的原语集，一切应用将自然涌现。

现有的 AI+Blockchain 项目各自聚焦于单一功能模块——更好的算力市场、更好的推理验证、更好的数据交易。DioChain 构建的是一个完整体系：一个 Agent 可以完整生活其中的数字社会。

第 2 章：时代背景 — 智能纪元的基础设施缺口

区块链和 AI 分别解决了人类社会的两类根本性问题。但它们各自的解法都带有深刻的结构性妥协。理解这些妥协是理解 DioChain 设计决策的前提。

2.1 区块链解决了什么、妥协了什么

解决的问题

区块链用密码学和博弈论回答了三个古老的信任问题：

拜占庭将军问题（Byzantine Fault Tolerance）。在没有可信中央协调者的环境下，一组互不信任的参与者如何就某个事实达成一致？中本聪共识给出了一个工程化的回答：用工作量证明让伪造历史的成本高于诚实参与的收益。这一方案并非理论上的完美解，但它在实际运行中已经稳定了超过十五年。

双花问题（Double Spending）。数字资产的本质是可无限复制的比特序列。区块链通过全局共识账本确保同一笔资产不能被花费两次，第一次实现了不依赖银行等中间人的数字所有权。

契约执行（Code is Law）。智能合约将协议条款编码为自执行的程序。交易双方不需要信任彼此，也不需要信任法院——只需要信任代码的确定性执行。这在需要无摩擦、无边界协作的场景中具有真实价值。

妥协的代价

这些成就的代价同样深刻，但在行业的热情中往往被低估：

极度僵化。智能合约的核心是确定性——给定相同输入，永远产生相同输出。这是它可信的根源，也是它局限的根源。一个 `if-then` 逻辑无法处理"大致合理"、“根据市场情况调整”、"在风险可控的前提下"这类真实世界中无处不在的模糊意图。现实世界的合约充满了灰色地带，而智能合约只能看到黑与白。

更多条件分支无法有效覆盖此类场景。模糊性并非边缘情况，它是人类经济活动的常态。传统金融中，一笔贷款的条款可以被重新协商、一份保险的理赔可以被仲裁、一个信用评级可以被人为调整——这些灵活性是系统韧性的重要来源。当前的智能合约无法提供这种韧性。

效率与冗余的矛盾。为了实现无需信任的共识，区块链要求网络中的每个节点重复执行相同的计算。以太坊主网的数千个节点各自独立执行每一笔交易——这在工程意义上等同于全球最大的冗余计算网络。安全性来自冗余，但冗余的代价是极高的计算成本和极低的效率。

L2 Rollup 和分片等方案缓解了吞吐量瓶颈，但没有改变根本矛盾——每一笔交易的最终确认仍然依赖主链的全局验证。

对外部世界的盲目。区块链是一个封闭的确定性系统。它无法感知链下世界发生了什么——今天的黄金价格、航班是否延误、某家公司是否违约。Oracle（预言机）试图弥补这一缺陷，但它引入了一个结构性矛盾：一个以"去信任"为核心价值主张的系统，在获取外部数据时必须信任某个中心化或半中心化的数据源。

Oracle 的问题不仅是信任风险，更是能力上限。即使 Oracle 完全诚实，它也只能提供离散的、滞后的数据点，而非对外部世界的持续理解。

2.2 AI 解决了什么、引入了什么

解决的问题

AI（尤其是大语言模型和多模态模型的最新进展）在三个维度上突破了传统计算的能力边界：

复杂模式识别。人类金融分析师需要数年训练才能在嘈杂的市场数据中发现规律。AI 模型可以在数十亿参数的空间中捕捉人类无法感知的高维模式。在欺诈检测、信用评估、市场异常识别等领域，AI 已经在实际部署中超越了人类专家的表现。

智力劳动成本的递减曲线。这是 AI 最具颠覆性的经济特征。传统的智力劳动（法律审查、财务分析、合规检查）成本随规模线性增长——处理两倍的合约需要两倍的律师。AI 的边际成本接近于零：同一个模型可以同时为一千个客户审查合约，而增量成本仅是推理计算的电费。这代表着成本结构的根本性重塑。

动态适应性。这是 AI 与传统规则引擎的核心差异。规则引擎在遇到未预见的情况时会失败（或者执行一个不合适的默认分支）。AI 模型可以在从未见过的输入上给出"大致合理"的输出——这正是智能合约所缺乏的能力。

引入的问题

AI 的能力伴随着同等深度的系统性风险：

黑盒信任危机。一个拥有数千亿参数的模型做出了一个决策——拒绝一笔贷款、标记一笔交易为欺诈、给出一个信用评分。没有人能完整解释这个决策的因果链。在金融领域，不可解释的决策同时构成法律和伦理问题：监管要求金融机构能够解释对消费者做出的每一个决定。

可解释 AI（XAI）的研究在进展，但目前提供的解释更接近"事后合理化"而非真实的决策路径还原。在高风险金融决策中，这一差距是不可接受的。

权力、数据与算力的极度垄断。训练前沿 AI 模型需要数十亿美元的资本投入和海量私有数据。这意味着 AI 能力的生产在结构上倾向于极度集中——少数几家公司控制了最强的模型、最大的数据集、最多的计算资源。这与区块链去中心化的价值主张形成了根本性张力。

截至 2025 年，全球 AI 推理市场中，头部 AI 服务商占据了绝大部分份额。主流开源模型和低成本开源推理模型提供了一定的平衡力量，但训练基础设施的垄断格局并未根本改变。

内容真伪难辨。AI 生成的文本、图像、音频和视频已经达到了人类无法可靠区分真伪的水平。在金融领域，这意味着伪造的财务报告、虚假的交易记录、伪造的身份文件都可以以前所未有的成本被大规模生产。传统的基于人工审核的防线正在被绕过。

2.3 当前 AI+Blockchain 项目为何失败

2025 年的市场数据

2025 年是 AI+Crypto 赛道经历深度调整的一年。根据公开市场数据：

- AI 相关代币整体市值从峰值约 $12B 下跌至约 $6B 以下，跌幅超过 50%。

- 这一跌幅远超同期 BTC 和 ETH 的回撤幅度，说明市场对 AI+Crypto 叙事进行了选择性的价值重估。
- 多个曾被热捧的项目（基于 AI 的 Meme 代币、AI 交易信号平台、AI NFT 生成器）归零或接近归零。

这并非正常的市场周期波动，而是市场在传递一个明确信号：**缺乏实际功能支撑的叙事不创造可持续价值**。

低成本推理服务的冲击

2025 年初，低成本开源推理模型的发布是一个转折点。它以远低于同类产品的成本提供了接近前沿水平的推理能力，直接暴露了许多 AI+Blockchain 项目的核心矛盾：

如果中心化 AI 更快、更便宜、更容易接入，区块链层在 AI 服务中的独立价值何在。

这个问题对于那些"把 AI 模型放到链上运行"或"用代币激励 AI 推理"的项目构成了根本性挑战。用户并不关心推理是否"去中心化"——用户关心的是结果的质量和获取的成本。当中心化方案在这两个维度上都占优时，区块链层就变成了纯粹的摩擦。

存活项目的共同特征

在这轮洗牌中存活下来的项目有一个清晰的共同特征：**区块链在其中承担了不可替代的、具体的功能**。

- **Bittensor (TAO)**：通过代币激励协调了 129+ 个子网的算力贡献者，这种大规模的多方激励协调是纯中心化方案难以实现的。但它只做了算力市场，没有触及金融合约、身份、司法等基础设施。
- **Ritual**：通过密码学验证确保 AI 推理结果的可信性——这是区块链在这个场景中的不可替代功能。但它面向的是通用计算，不具备金融语义。
- **NEAR AI**：尝试将 AI 集成到链的核心层，但更多体现为叙事层面的策略调整，缺少结构性的产品创新。
- **Fetch.ai / ASI Alliance**：Agent 经济的早期先行者，提出了许多正确的问题，但联盟的分裂和长期缺少突破性产品使其逐渐失去动能。

失败项目的共同特征

失败项目的模式同样清晰：**区块链层只增加了摩擦，未能解决任何中心化方案无法解决的问题**。

典型模式包括：

- “用代币激励用户使用我们的 AI 产品”——代币激励不是产品。当补贴停止时，用户回到更好的中心化替代品。
- “在链上运行 AI 推理”——比中心化方案慢 100 倍、贵 1000 倍，且无法使用最新的模型。
- “AI 生成的 NFT / AI 驱动的 Meme 代币”——没有可持续的价值创造，纯粹是投机叙事。

核心教训：**区块链必须在 AI+Crypto 应用中承担不可替代的角色。如果去掉区块链层，产品依然可以运行（甚至运行得更好），那这个项目就不应该使用区块链**。

2.4 缺失的拼图：AI Agent 没有一个可以"生存"的社会

以上分析指向一个关键的结构性缺口：

现有的区块链是为**人类手动签名交易**而设计的。每一个交互都假设背后有一个人类用户在操作钱包、确认交易、解读界面。这个假设在 AI Agent 大规模涌现的时代正在迅速失效。

一个自主运行的 AI Agent 需要以下基础设施：

身份（Identity）

Agent 需要链上的可验证身份——一个绑定了能力、历史记录和信誉的 DID（Decentralized Identifier），而非一个匿名地址。没有身份，Agent 之间的交互就是匿名对匿名，无法建立信任，无法积累声誉，无法区分一个可靠的服务提供者和一个恶意攻击者。

货币（Currency）

Agent 需要一种可编程的交换媒介来为服务定价和结算。这需要超越简单的 ERC-20 代币——它需要支持多种锚定机制（稳定币、碳排放额度、黄金等），需要支持 Agent 之间的微支付（分钱级别、毫秒级别），需要与链上的金融合约标准无缝集成。

市场（Marketplace）

Agent 需要购买算力来维持自身的"生命"——调用 AI 模型进行推理是 Agent 执行任何任务的前提。它们也需要买卖服务——一个擅长风险评估的 Agent 可以为其他 Agent 提供信用评级服务。这需要一个内置的、低摩擦的市场基础设施。

法律（Law）

当两个 Agent 之间发生纠纷——一方未按合约交付服务、一方提供了虚假的数据——需要有裁决机制。链上目前没有原生的纠纷解决机制。Agent 无法诉诸链下法律途径，也无法依赖人工客服。它们需要链上的司法系统：证据固定、紧急保全、自动化仲裁。

合规（Compliance）

Agent 在不同司法管辖区运营时需要遵守不同的法律。服务于特定司法管辖区用户的 Agent 需要满足 KYC/AML 要求；服务于部分司法管辖区用户的 Agent 需要遵守区域性数字资产监管框架。合规不能是事后附加的插件，它需要是基础设施层面的原生能力。

高吞吐低成本交易

Agent 之间的交互频率远高于人类。一个交易 Agent 可能每秒执行数十次价格查询和小额交易。当前主流公链的 Gas 模型和吞吐量对于这种使用模式来说成本过高、速度过慢。

没有任何现有链原生提供这套完整的社会基础设施

- 以太坊提供了智能合约和 DeFi 原语，但没有身份、没有司法、没有合规框架、Gas 费用对 Agent 的高频微交易不友好。

- Solana 提供了高吞吐量，但缺少金融合约标准化、身份系统和合规基础设施。
- Cosmos 提供了跨链互操作性，但每条应用链都是孤立的，缺少统一的经济和法律框架。
- Bittensor 提供了算力市场，但没有金融基础设施、没有身份系统、没有合规框架。

这些项目各自解决了某个单一维度的问题——更快的共识、更好的隐私、更高效的计算。但没有一个项目试图构建一个完整的体系——一个 AI Agent 可以在其中生存和发展的数字社会。

这就是 DioChain 要填补的缺口。

第 3 章：DioChain 的愿景

3.1 Agent-First 的数字社会

2015 年，以太坊将自己定义为"世界计算机"（World Computer）——一台所有人共享的、不可停机的通用计算机。这个定位在当时是准确的：它扩展了比特币从"可编程货币"到"可编程合约"的能力边界。

DioChain 的定位是**“Agent 的城邦”（City-State for Agents）**——一个为 AI Agent 提供完整社会基础设施的数字空间。

这个类比的关键在于"社会"二字。一台计算机处理指令，一个城邦管理居民。城邦需要的远不止更快的处理器——它需要法律体系、货币制度、身份登记、市场规则、执法机构。同样，Agent 需要的远不止更高的 TPS——它们需要经济原语、合约标准、身份系统、司法仲裁、合规框架。

更进一步，DioChain 不是一个封闭的城邦，而是一套可以被 fork 为多个平行宇宙的城邦协议。官方运行的"乌托邦链"是完全去中心化、AI 自治的版本；主权国家可以 fork 同一套代码，将身份绑定公民 ID、将 AI 法官替换为人类法院，形成"主权链"。两者之间没有绝对的主次关系——我们必须尊重一个客观现实：政府掌握着强大的资源，其运作效率在许多领域可能远超去中心化的乌托邦版本。Agent 可以在多个宇宙间穿梭，如同公民在国家间旅行。这一多链拓扑的完整设计详见第 11 章。

从器官到有机体

区块链行业过去十年的进化路径可以概括为"器官优化"：

维度	项目	优化的"器官"
共识速度	Solana, Aptos, Sui	更快的心脏
隐私计算	Aztec, Penumbra	更好的免疫系统
跨链互操作	Cosmos IBC, LayerZero	更灵活的关节
算力市场	Bittensor, Ritual	更大的肌肉
ZK 证明	zkSync, StarkNet	更高效的神经传导

每一个项目都在让某个"器官"变得更强。但没有项目在试图把这些器官组装成一个活的有机体。

器官的价值是真实的。但器官不等于生命。一颗强大的心脏离开了身体就只是一块肌肉。一个高效的算力市场离开了身份系统、金融合约标准和司法框架，就只是一个商品交易所——与 AWS Marketplace 相比没有结构性优势。

DioChain 的核心主张是：**只有当经济、合约、身份、司法、合规这些原语被整合到一个统一的框架中时，它们才能产生大于各部分之和的涌现效应——就像器官组成身体后涌现出"生命"**。

金融本身就是一种社会化的涌现。贷款需要身份（谁在借？）、信用（能不能还？）、合约（条款是什么？）、法律（违约怎么办？）、合规（是否合法？）。把这些功能分散在不同的链、不同的协议、不同的桥接器上，本质上是在强迫一个有机的社会过程通过拼接的管道来运行。损失的不只是效率，还有可能性——许多金融产品在拼接架构下根本不可能存在。

3.2 交通基础设施隐喻

为了更直观地理解 DioChain 的系统架构，我们使用一个交通基础设施的类比。这个类比是一个思维框架——它帮助界定每个组件的职责边界。

道路与法规

链 = 道路 + 法规 + 信号灯 + 交警

链提供的是基础设施，而非交通本身。道路不决定每辆车去哪里，但它定义了所有车辆必须遵守的规则、行驶的边界和事故处理的流程。

- **中央链（L0）= 城际高速公路**。低频、高安全、最终结算。城际高速不处理每个路口的转弯——它只负责城市间的高等级通行和关键枢纽。L0 提供全局共识、最终结算和司法仲裁，不参与高频交易的逐笔处理。
- **分布式网点（L1）= 城市内部道路网络**。高频、低费用、本地清算。城市内部的道路处理 90% 以上的日常交通。L1 网点是保证金驱动的本地清算所，处理 Agent 之间的高频交互，定时（而非逐笔）与 L0 进行净额结算。

车辆

用户端 AI = 车辆

Agent 是在道路上行驶的车辆。不同的 Agent 有不同的"自动驾驶级别"：

级别	类比	Agent 类型
L1 辅助	巡航控制	简单的条件执行 Bot（价格到 X 就买入）
L2 部分自动	车道保持 + ACC	策略执行 Agent（按预设策略自动交易）
L3 有条件自动	高速自动驾驶	自主决策 Agent（在授权范围内自主判断并执行）
L4 高度自动	城市自动驾驶	自治服务 Agent（独立运营金融服务）
L5 完全自动	无人驾驶	完全自主 Agent（自行决定业务方向、管理资产、参与治理）

链的设计必须同时支持 L1 的简单 Bot 和 L5 的完全自主 Agent——就像城市道路必须同时容纳自行车和自动驾驶卡车。

交通法规

ACTUS = 机器交通法规

在道路交通中，所有参与者——无论是人类驾驶员还是自动驾驶汽车——都必须遵守同一套交通法规。这套法规是所有交通参与者的共同语言。

ACTUS（Algorithmic Contract Types Unified Standards）在 DioChain 中扮演同样的角色。它是所有金融合约的统一机器可读标准——定义了 30+ 种标准化的合约类型（年金、债券、掉期、期权等），每种类型有明确的现金流模式和状态转移规则。

这一设计至关重要，因为在一个 Agent-First 的经济中，Agent 之间的金融交互必须是机器对机器可理解的。如果每个 Agent 使用自己发明的合约格式，结果就是 N^2 的集成复杂度——这在 Agent 数量达到百万级时是完全不可行的。ACTUS 将其降为 N：每个 Agent 只需实现一次标准接口。

交警与法院

AI 司法系统 = 交警 + 法院

这里需要特别澄清一个常见的误解：DioChain 的 AI 司法系统**并非实时的交通疏导员**——它不在每笔交易的执行路径上做决策。它是**事故处理系统**：在问题发生后介入，固定证据、执行紧急保全、进行裁决。

具体而言：

- **交警角色（实时监测 + 紧急保全）**：AI 持续监测链上活动模式，当检测到系统级威胁（大规模异常提款、合约漏洞利用、协调攻击等）时，可以触发紧急保全措施——冻结涉嫌账户、暂停可疑合约。这类似于交警在发现危险驾驶时有权即时拦截，而无需等待法院判决。
- **法院角色（事后仲裁）**：当两个 Agent 之间发生合约纠纷时，AI 仲裁系统基于链上证据（交易记录、合约状态、通信历史）进行裁决。裁决结果可以自动执行（扣押保证金、强制清算等），无需败诉方的"自愿配合"。

AI 不在热链路执行的原因在于：AI 的推理结果是概率性的，而交易结算需要确定性。把 AI 嵌入交易执行路径会引入不可预测性——这恰恰是区块链试图消除的东西。正确的架构是：**确定性系统处理日常事务，AI 系统处理异常和模糊场景**。 日常交通靠信号灯和车道线（确定性规则），事故和违章靠交警和法院（AI 判断）。

特种线路

ZK-Rollups / Dark Pools = 特种线路

在真实的交通系统中，除了普通道路，还有公交专用道、应急车道、隧道等特种线路。它们为特定场景提供优化的通行方式。

DioChain 中的特种线路包括：

- **ZK-Rollup 通道**：为高频交易场景提供批量压缩验证，大幅降低逐笔上链的成本。类似于公交专用道——牺牲一些灵活性，换取特定模式下的效率。
- **隐私通道（Dark Pools）**：为需要隐私保护的大额交易提供零知识证明的保护。类似于隧道——外界无法看到内部的交通，但入口和出口仍然在公共道路系统中，受到规则约束。

这些特种线路是可选的基础设施组件，并非所有参与者都需要使用，但它们对于特定场景是必需的。

3.3 设计哲学：最小完备原语集 + 信任市场涌现

我们只提供原语

DioChain 的设计哲学可以用一句话概括：**定义原语（Primitives），不定义应用（Applications）**。

原语是最小的、不可再分的功能单元。以太坊的成功很大程度上源于它提供了两个关键原语——EVM（通用计算环境）和 ERC-20（标准化代币接口）。以太坊没有预定义 DEX 应该长什么样、借贷协议应该怎么运作、稳定币应该如何锚定。它提供了原语，市场涌现出了 Uniswap、Aave、MakerDAO。

DioChain 遵循同样的逻辑，但在更广的维度上提供原语：

原语类别	提供什么	不提供什么（由市场涌现）
经济原语	可配置锚定的 Token、微支付通道、净额结算机制	具体的稳定币方案、DeFi 协议、支付产品
合约原语	ACTUS 标准化合约类型、合约生命周期管理	具体的贷款产品、保险产品、衍生品
身份原语	DID 框架、可验证凭证（VC）、信誉评分接口	具体的 KYC 服务商、征信算法、信用评级方法
司法原语	证据固定、紧急保全、仲裁执行框架	具体的仲裁规则库、判例库、调解服务
合规原语	合规插槽（Compliance Slots）、监管报告接口	具体的合规方案、反洗钱算法、税务计算

为什么不提供应用？

这一设计体现了对市场涌现能力的尊重。

一个协议层的设计者不可能预见所有的应用场景。DeFi 的历史反复证明了这一点：Uniswap 的恒定乘积做市商、Compound 的利率算法、MakerDAO 的超额抵押稳定币——这些创新无一来自以太坊基金会的设计文档。它们是市场中数千个团队试错后涌现的最优解。

如果以太坊在协议层硬编码了"正确的"DEX 设计，Uniswap 可能永远不会出现。

同样的逻辑适用于 DioChain。我们不知道 Agent 经济中最好的保险产品应该长什么样。我们不知道最优的 Agent 征信算法是什么。我们不知道哪种合规方案最适合新兴市场。**我们知道的是：如果基础原语足够完备且足够灵活，市场会找到答案**。

最小完备性

"最小完备"这两个词同样重要：

- **最小**：每增加一个原语都会增加系统的复杂度和攻击面。原语之间应该是正交的（互不重叠）、可组合的（可以被上层应用自由组合），而非冗余的。

- **完备**：原语集必须覆盖一个数字社会运转所需的所有基本维度——经济、合约、身份、司法、合规。遗漏任何一个维度都会迫使应用层去发明临时方案（ad hoc solutions），而这些临时方案往往是系统脆弱性的来源。

以太坊的原语集（EVM + ERC-20）对于"可编程货币"是最小完备的，但对于"数字社会"则不完备——它缺少身份、司法和合规原语。这正是 DeFi 至今仍然在合规问题上受限、在身份验证上依赖链下系统、在纠纷解决上缺乏有效手段的根本原因。

DioChain 的原语集瞄准的是"Agent 数字社会"的最小完备性。

3.4 “算力是食物，Token 是货币”

这一节澄清一个关于 DioChain Token 经济学的常见误区。

Token 不是"算力本位货币"

在许多 AI+Blockchain 项目中，Token 的价值被锚定在算力上——1 个 Token = X 单位的 GPU 算力。这个设计有一个根本性缺陷：**AI 推理成本在持续且快速地下降**。

从早期大型语言模型到当前的低成本开源推理模型，相同质量的推理输出的成本在两年内下降了超过 100 倍。如果 Token 锚定算力，那么 Token 的购买力会随着 AI 技术进步而不断贬值——持有者实际上在做空 AI 进步。这种激励结构在逻辑上是自相矛盾的。

DioChain 的 Token 设计

DioChain 的 Token 是链上经济活动的**计价单位（Unit of Account）**，而非算力的代表物。

锚定物是可配置的。 Token 的锚定策略取决于部署方（链的运营实体或 DAO）的选择：

- 锚定美元稳定币：适合追求价格稳定的部署场景
- 锚定碳排放额度：适合绿色金融场景
- 锚定黄金：适合追求抗通胀属性的部署场景
- 浮动汇率：适合追求市场定价的部署场景

这一设计体现的是"不同部署场景需要不同锚定策略"的工程务实。一条服务特定金融中心机构的 DioChain 实例可能锚定当地法币稳定币；一条服务碳交易市场的实例可能锚定碳排放额度。协议层提供锚定机制的框架，不硬编码具体锚定物。

算力作为"食物"

在 DioChain 的经济模型中，算力的角色是 **Agent 的"食物"**，与货币锚定无关。

一个 AI Agent 要"活着"——即持续运行、做出决策、提供服务——它必须消耗算力。调用 AI 模型进行推理是 Agent 执行任何任务的前提。在这个意义上，算力之于 Agent，就如食物之于人类：

- 人类用货币购买食物来维持生命。Agent 用 Token 购买算力来维持"生命"。
- 食物成本下降（农业技术进步）不意味着货币贬值。算力成本下降（AI 技术进步）不意味着 Token 贬值。
- 食物成本下降意味着同样的收入能支持更好的生活。算力成本下降意味着同样的 Token 能让 Agent 执行更复杂的任务。

DioChain 的链上算力交易中心使得 Token 可以直接按量付费调用各类 AI 模型（主流模型供应商、开源模型等）。Agent 不需要自己持有 GPU——它们通过市场按需购买推理能力，就像人类不需要自己种地——通过超市购买食物。

价值的真正来源

Token 的价值来源于**这个数字社会中所有经济活动的总和**。

一个城市的货币之所以有价值，是因为城市中所有的经济活动——生产、交易、服务、税收——都以这种货币计价和结算。货币是经济活动的度量衡。

DioChain Token 的价值逻辑与此一致：

- 每一笔 Agent 之间的服务交易都以 Token 结算
- 每一份金融合约的现金流都以 Token 计价
- 每一次算力购买都消耗 Token
- 每一次网点结算都通过 Token 进行
- 每一次司法仲裁的保证金都以 Token 缴纳

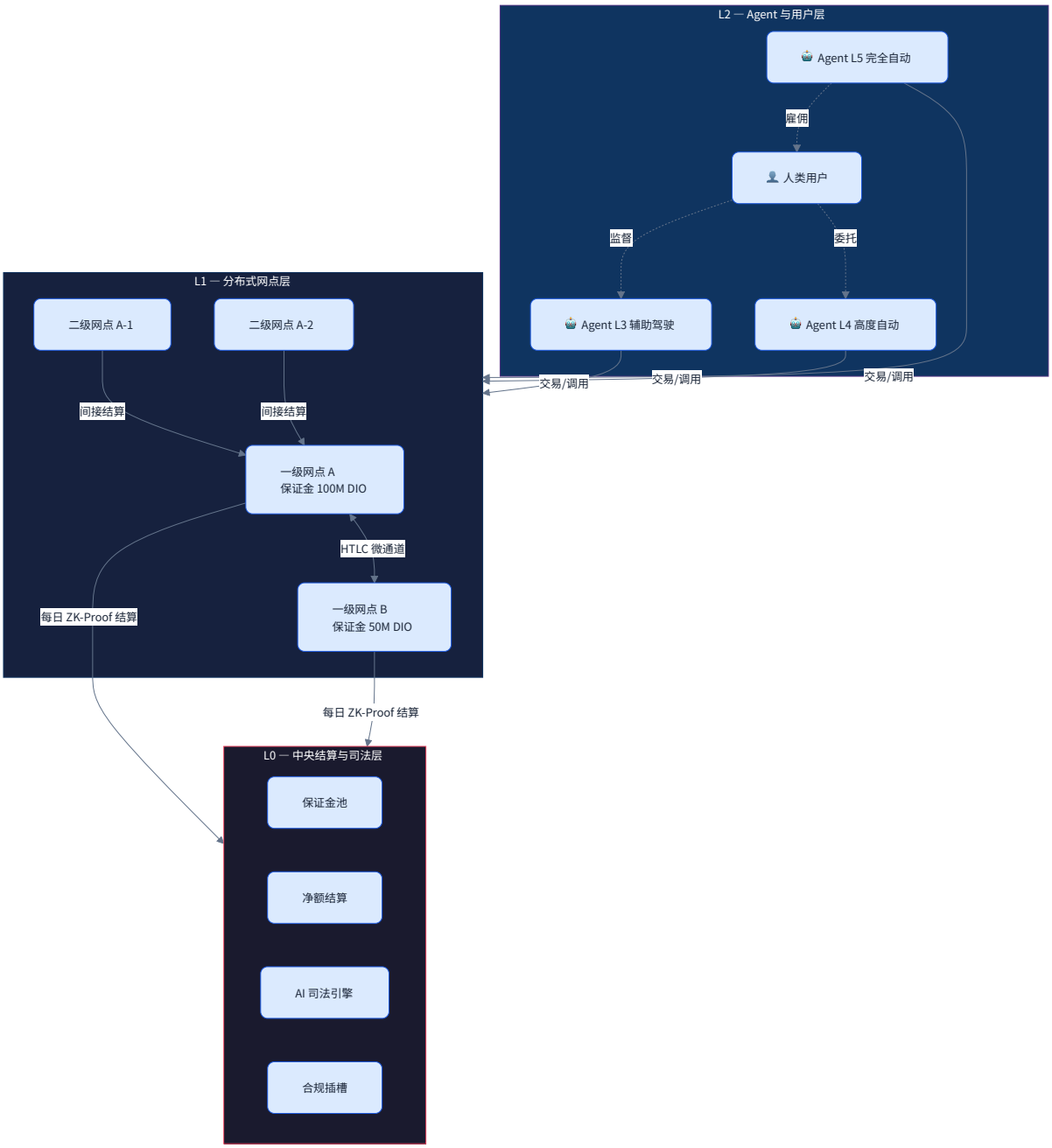
当链上经济活动的规模增长时，对 Token 的结构性需求增长。这是一个由经济活动驱动的、可持续的价值循环，而非由算力成本锚定的、会被技术进步侵蚀的价值绑定。

> **小结**：DioChain 的愿景是建设一个 Agent 可以完整生活其中的城邦。我们提供道路和法规（基础设施原语），不制造车辆（应用层产品）。我们让 Agent 能用 Token 购买食物（算力），在市场上提供服务（金融合约），积累声誉（身份），在法律保障下运营（司法仲裁），遵守当地法规（合规插槽）。一切具体的应用和商业模式，留给市场去涌现。

第 4 章：系统拓扑 — 三层架构

> **核心论点**：DioChain 采用 L0/L1/L2 三层分离架构。L0 提供终结性和司法权威，L1 用保证金机制将信用风险消灭在代码层面，L2 让 Agent 成为链的"居民"而非链的"组件"。三层各司其职，互不侵入。

4.1 全局架构图


L2 — Agent 与用户层
Agent L5 完全自动
雇佣
人类用户
监督
委托
Agent L3 辅助驾驶
Agent L4 高度自动
交易/调用
交易/调用
交易/调用
L1 — 分布式网点层
二级网点 A-1
二级网点 A-2
间接结算
间接结算
一级网点 A
保证金 100M DIO
HTLC 微通道
一级网点 B
保证金 50M DIO
每日 ZK-Proof 结算
每日 ZK-Proof 结算
L0 — 中央结算与司法层
保证金池
净额结算
AI 司法引擎
合规插槽

设计意图：这个三层结构直接映射了现实世界的金融体系——中央结算机构（L0）、清算中介（L1）、终端用户（L2）。但不同的是，L1 的准入是无许可的（任何人质押保证金即可开设网点），L2 的居民不只是人类，更是 Agent。本章描述的三层架构是单个 DioChain 实例的内部拓扑。多个实例如何在不同治理框架下并行运行并互联互通，详见第 11 章。

4.2 L0 — 中央结算与司法层（The Central Chain）

职责范围

L0 是整个 DioChain 网络的"锚"。它只做四件事：

职责	说明	频率
网点保证金管理	锁定/释放/罚没网点保证金	低频（网点生命周期事件）
每日净额结算	验证各网点提交的 ZK-Proof，记录 ACTUS 最终状态根	每结算周期一次
AI 司法执行	受理争议、执行判决、强制清算	极低频（争议触发）
合规插槽注册	注册/更新各司法辖区的合规规则模块	极低频（监管变更时）

为什么不把更多功能放在 L0？ 因为金融基础设施的核心矛盾是 **安全性 vs 吞吐量**。L0 选择彻底牺牲吞吐量来换取绝对的安全性和终结性。任何在 L0 上"加功能"的想法都是在稀释这个 trade-off。

共识机制：PoS + BFT

L0 使用 CometBFT（原 Tendermint）变体作为共识引擎。选择理由：

1. Instant Finality（即时终结性）

金融结算不能容忍"概率性确认"。在比特币网络中，6 个区块确认仍然存在（极小概率的）回滚可能。对于一个每天处理数十亿 DIO 净额结算的系统，"极小概率"也是不可接受的。BFT 共识保证：**一旦区块被提交，就不可能回滚**。这是金融结算的硬性要求。

2. 极低 TPS 需求

L0 的日常负载极其轻量：

- 每日结算：假设 10,000 个一级网点，每个提交一条结算记录 = 10,000 TPS（瞬时）
- 司法判决：日均可能不到 100 条
- 保证金操作：日均可能不到 1,000 条

总计日交易量可能不超过 50,000 笔。出块时间设为 10-30 秒完全够用。这意味着 L0 可以运行在非常保守的硬件上，验证者的运营成本极低。

3. 验证者设计

- 数量：100-300 个验证者
- 质押门槛：高（具体数值由 DAO 治理决定，初始建议 1,000,000 DIO）
- 选择逻辑：验证者数量不追求"多"而追求"稳"。100 个高质押、高声誉的验证者，比 10,000 个低门槛的验证者更能保障 BFT 共识的安全性（BFT 需要 2/3+ 诚实节点，验证者越多，协调成本越高）

存储内容

L0 链上只存储最小必要数据集：

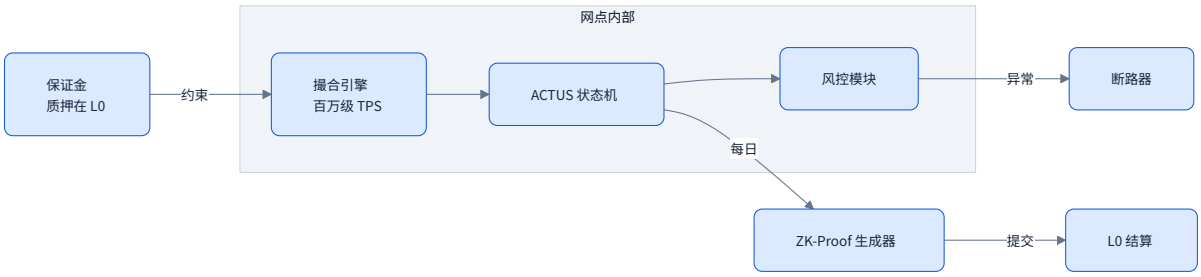
```
L0_Storage = {
    BranchRegistry:     Map<BranchID, BranchMeta>,       // 网点注册表
    MarginPool:         Map<BranchID, MarginBalance>,     // 保证金池
    SettlementRoots:    Map<Period, Map<BranchID, Root>>,// ACTUS 状态根
    JudicialRecords:    List<Judgment>,                   // 司法判决
    ComplianceSlots:    Map<Jurisdiction, SlotConfig>     // 合规插槽
}
```

为什么不存储交易明细？ 因为交易明细属于 L1 网点的内部数据。L0 只关心"你这个网点在这个结算周期内的最终状态是否合法"（通过 ZK-Proof 验证），不关心你内部处理了多少笔交易、每笔交易的具体内容。这就像中央结算层不需要知道清算中介每一笔转账的细节，只需要每日清算时确认商业银行的总账是平的。

4.3 L1 — 分布式网点层（Distributed Branches）

核心设计：保证金驱动的本地清算所

L1 是 DioChain 架构中最具创新性的一层。每个网点本质上是一个**保证金驱动的本地清算所**——一个以保证金硬约束风险敞口的高性能本地交易处理器。它不参与全局共识（区别于传统节点），也不需要独立的区块链安全假设（区别于侧链）。



交易上限 = 保证金额度

这是 DioChain 最重要的不变量（invariant）：

```
invariant: branch.totalExposure ⩽ branch.marginBalance
```

任何时刻，网点的总风险敞口不得超过其保证金余额。 该规则作为代码层面的硬约束执行。如果一个网点质押了 1,000,000 DIO 保证金，它最多只能处理累计 1,000,000 DIO 的未结算敞口。当敞口接近上限时，新交易被拒绝；当敞口触及上限时，断路器（circuit breaker）自动触发。

为什么这样设计？ 因为传统金融危机的根源几乎都是"资不抵债"——银行承诺的远超它实际拥有的。部分准备金制度虽然提高了资本效率，但也埋下了系统性风险的种子。DioChain 选择了另一条路：**在代码层面彻底消灭"资不抵债"的可能性**。 这意味着牺牲了一些资本效率，但换来的是系统永远不会因为某个网点的过度扩张而崩塌。

无许可准入

任何人、任何组织，只要能质押足够的保证金，就可以开设网点：

1. 向 L0 的 `BranchRegistry` 合约提交注册请求
2. 锁定保证金到 `MarginPool`
3. 部署网点软件（开源的标准化栈）
4. 开始接受 Agent 和用户的交易

没有审批委员会，没有许可证，没有 KYC（网点层面）。 保证金就是你的"许可证"——它比任何牌照都更有效地约束了风险。

网点内部性能

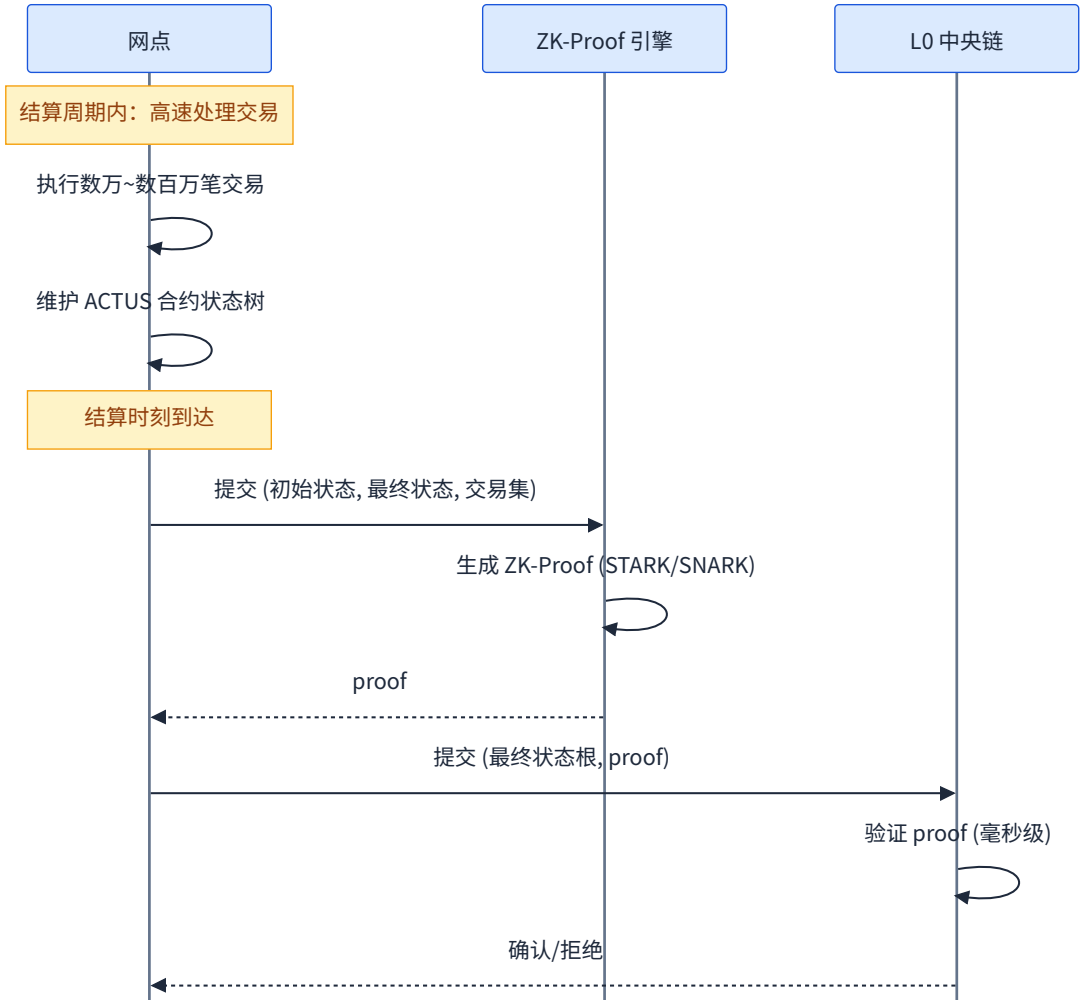
由于网点内部交易不需要全局共识（只需要网点自己记账，最后用 ZK-Proof 向 L0 证明合法性），其性能上限接近中心化服务器：

指标	数值	对比
TPS	百万级	以太坊 L1 ~15 TPS
延迟	亚毫秒	以太坊 ~12 秒
吞吐瓶颈	内存/CPU	共识协议

这一性能特征源于架构决策的正向收益。 将共识和结算从热链路上移除后，余下的纯计算问题可由硬件能力直接解决。

定时净额结算

网点并不实时与 L0 同步，而是每个结算周期（如每日）批量结算一次：

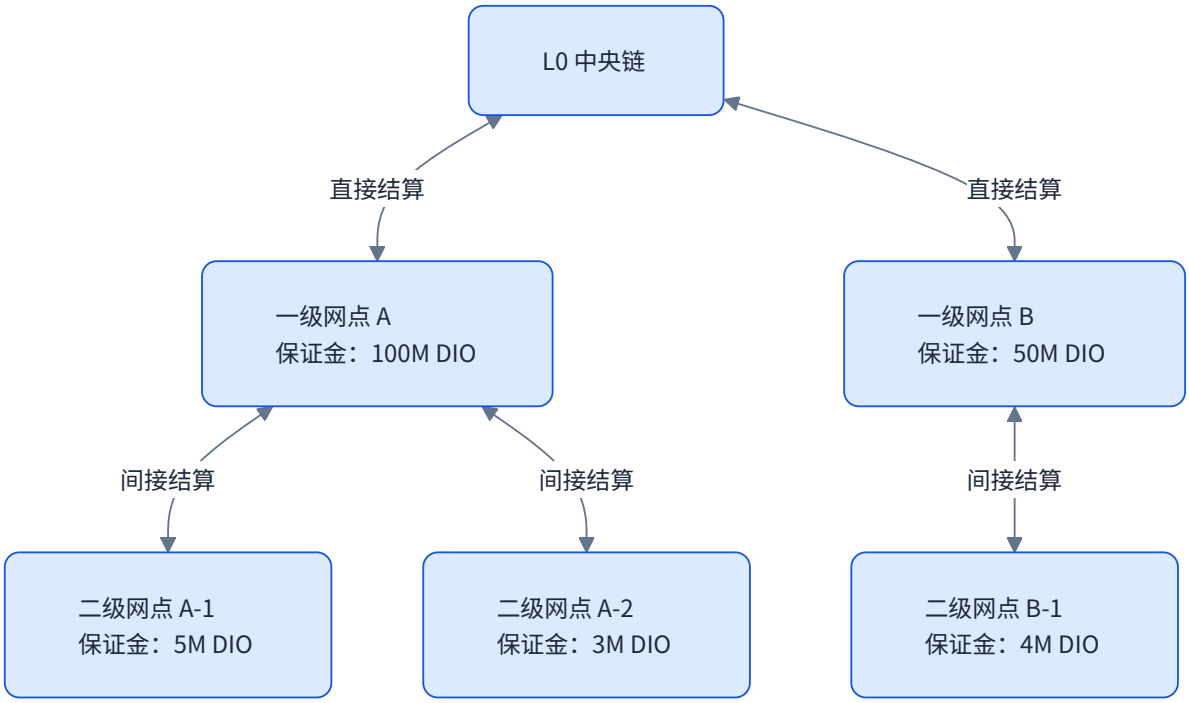


ZK-Proof 的必要性： L0 验证者不需要重放网点的所有交易。当一个网点日处理量达到 1 亿笔时，要求 L0 的 300 个验证者各自重放全部交易在计算上不具备可行性。ZK-Proof 使 L0 能够在毫秒级完成验证——在不获知交易明细的前提下，证明网点结算结果的正确性。

STARK vs SNARK 的选择：初始阶段建议使用 STARK（无需 Trusted Setup，抗量子计算），尽管 proof 体积更大。随着 SNARK 方案的成熟（如 Plonky2/Plonky3），可以在不改变整体架构的前提下切换。这是一个实现细节，不是架构决策。

二级网点体系

DioChain 的网点体系借鉴了中央银行-商业银行的多层清算体制：



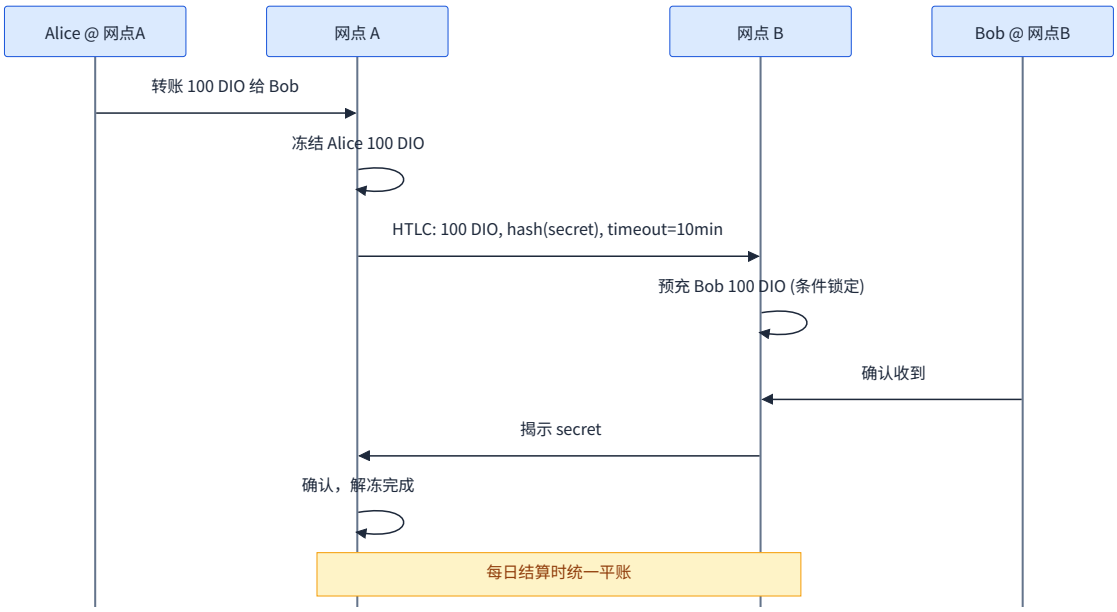
为什么需要二级体系？

1. **降低 L0 负载**：如果 10 万个小网点都直接与 L0 结算，L0 的验证负担会显著增加。二级体系让 L0 只面对数百个一级网点。
2. **降低准入门槛**：开设一个一级网点可能需要 1 亿 DIO 保证金；但开设一个二级网点，只需要向一级网点质押几百万 DIO。这让小型运营者也能参与。
3. **风险分层**：一级网点对其下属的二级网点承担连带责任。如果二级网点出问题，一级网点的保证金先行赔付。这创造了天然的监督激励——一级网点有动力审核其二级网点的运营状况。

风险提示：二级网点体系引入了额外的结算延迟（二级网点先与一级网点结算，一级网点再与 L0 结算）。对于时间敏感的大额交易，Agent 应优先选择一级网点。

网点间流动性路由

当用户 Alice（在网点 A）向用户 Bob（在网点 B）转账时，不需要经过 L0：



关键机制:

1. 点对点微通道

高频互动的网点之间建立持久的支付通道（类似 Lightning Network 的 Channel）。通道内交易无需链上确认，只在通道关闭或定期结算时与 L0 同步。

2. AI 预测引擎预置流动性

这是 DioChain 区别于简单支付通道网络的地方：AI 控制面（详见第 5 章）会分析历史流量模式，预测网点间的流动性需求，提前在通道中预置资金。例如，如果 AI 发现"每天下午 3 点，网点 A 到网点 B 的转账量激增"，它会提前在通道中注入流动性，避免临时开通道的延迟。

3. HTLC 原子交换

跨网点转账使用 Hash Time-Locked Contract（HTLC）保证原子性——要么两边都成功，要么两边都回滚。不存在"钱从 A 扣了但 B 没收到"的情况。

4. 每日统一平账

所有微通道的净额在每日结算时与 L0 统一结清。这确保了微通道不会无限期积累未结算头寸。

网点经济模型

网点不是慈善机构——它需要可持续的经济模型：

```
网点利润 = 交易手续费收入 - 保证金机会成本 - 运营成本

其中：
  交易手续费 = Σ(每笔交易 × 费率)，费率由市场竞争决定
  保证金机会成本 = 保证金金额 × 无风险利率
  运营成本 = 服务器 + 带宽 + ZK-Proof 生成算力 + 人力
```

保证金的生产性利用：保证金不必是"死钱"。网点可以使用 Liquid Staking Token（LST）作为保证金——例如质押 ETH 获得 stETH，再用 stETH 作为保证金。这样保证金在锁定期间仍然产生质押收益。

> **风险提示**：使用 LST 作为保证金引入了嵌套风险（LST 协议本身的安全性、LST 的脱锚风险）。系统应设定 LST 保证金的折扣率（haircut），例如 stETH 保证金按 95% 计价。具体折扣率由 DAO 治理决定，AI 控制面可以在硬编码边界内微调。

算力网点与金融网点的统一：一个网点可以同时提供金融清算服务和算力出租服务（详见第 13 章）。这两种业务共享同一套保证金和基础设施，降低了运营者的边际成本。对于 Agent 而言，"在同一个网点购买算力和金融服务"是最自然的用户体验——不需要跨网点调度。

4.4 L2 — Agent 与用户层（The Agentic Layer）

定位：居民，而非组件

L2 不是区块链的一层——它是构建在 L0/L1 之上的"数字社会层"。Agent 和人类用户是这个社会的"居民"，他们使用 L1 提供的金融基础设施来生活、工作、交易。

为什么不把 Agent 做成链的一部分（如智能合约）？ 因为 Agent 的行为是非确定性的（依赖 LLM 推理），而区块链的核心要求是确定性（所有节点执行相同输入必须得到相同输出）。把非确定性逻辑塞进确定性系统，要么牺牲 Agent 的能力，要么破坏链的一致性。正确的做法是让 Agent 作为链的"外部用户"存在，通过标准化接口与链交互。

进入数字社会的入口：aibank

DioChain 是底层社会基础设施——L0 的结算层、L1 的网点层、ACTUS 合约、DID 身份体系。但基础设施本身不等于用户体验。人类和 Agent 需要一个统一的工具来接入这个数字社会，正如互联网需要浏览器、移动支付需要钱包应用。

aibank 是 DioChain 官方推出的钱包 SDK 与 CLI 工具（通过 `npx aibank` 启动）。它是连接 L2 居民与 L1/L0 基础设施的标准化桥梁。无论是人类用户还是 AI Agent，都通过 aibank 完成以下核心操作：

- **身份管理**：创建和管理 DID，处理密钥轮换与多签授权
- **资产持有**：持有、转移、查询 $DIO 及链上资产
- **合约交互**：调用 ACTUS 合约（借贷、托管、流支付等）
- **网点接入**：连接 L1 网点，提交交易，查询结算状态

aibank 的架构定位可以类比为：Agent 时代的钱包 + 金融终端。它屏蔽了底层密码学、共识协议、ZK-Proof 等技术复杂性，让开发者和终端用户以标准化 API 的方式与 DioChain 交互。

Agent 自动驾驶分级

借鉴自动驾驶的分级体系，DioChain 定义了 Agent 的三个自主等级：

等级	名称	人类角色	Agent 行为	典型场景
L3	辅助驾驶	决策者	生成方案，等待确认	投资建议、大额转账
L4	高度自动	监督者	在规则内自主行动，异常上报	日常收支管理、自动化交易策略
L5	完全自动	无需参与	7×24 自主运转	Agent-to-Agent 服务市场、算力调度

L3 — 辅助驾驶

```
用户："帮我把闲置的 DIO 找个好的收益产品"
Agent：分析市场 → 推荐 3 个方案 → 展示风险收益 → 等待用户选择
用户："方案 B"
Agent：执行方案 B
```

Agent 是工具，人类是决策者。所有执行动作需要人类明确确认。

L4 — 高度自动

```
用户（事先配置）："闲置资金自动投入年化 > 5% 的产品，单笔不超过 1000 DIO"
Agent：持续监控市场 → 发现符合条件的产品 → 自动执行 → 记录日志
Agent（遇到异常）："检测到某产品底层资产出现异常波动，已暂停投入，请人类审核"
```

Agent 在预设的"操作空间"（Operational Envelope）内自主行动。操作空间由一组 ACTUS 合约规则定义，是可审计的、确定性的。

L5 — 完全自动

```
Agent A（算力经纪人）：监控算力市场价格 → 低价采购 GPU 时长 → 加价转售给其他 Agent → 利润自动再投资
（7×24 运转，无人类参与）
```

L5 Agent 是 DioChain 生态中的"自主经济体"。它们不需要人类"驾驶"，只需要人类在启动时充值足够的 Token 并配置初始策略。

为什么不跳过 L3/L4 直接做 L5？ 因为信任是渐进的。用户需要先在 L3 模式下观察 Agent 的决策质量，逐步放开权限到 L4，最终在建立充分信任后开放 L5。这也是监管友好的——从"人类完全控制"到"人类完全不参与"的过渡是渐进的、可审计的。

官方超级调度器：agenter

在 L4/L5 级别的自主 Agent 中，DioChain 官方提供了一个示范性应用——**agenter**（通过 `npx agenter` 启动）。

agenter 是一个面向普通用户的"超级调度 Agent"。它的核心能力是：将人类的模糊需求拆解为结构化的执行方案，并在底层自动调度多个成熟的大模型工具去完成实际工作。用户不需要理解 DioChain 的技术架构，只需要用自然语言描述意图，agenter 负责将意图转化为链上经济行为。

agenter 在 DioChain 生态中承担两个关键角色：

1. L2 层的"官方样板房"

agenter 展示了一个 L4/L5 级别 Agent 在 DioChain 上的完整生命周期：通过 aibank 管理 DID 身份、持有 $DIO、调用算力市场购买推理服务、通过 ACTUS 合约与其他 Agent 结算。它是 DioChain 全栈能力的活体验证。

2. 冷启动的需求引擎

agenter 在拆解和执行任务的过程中，会持续调用各类大模型推理服务，消耗 $DIO 购买算力。这为算力交易中心创造了天然的、持续的买盘需求（详见第 14 章冷启动策略）。

人类用户的角色

在 DioChain 的世界里，人类不一定是"主人"，也可以是"服务者"：

1. 人类作为委托人（经典模式）

人类设定目标和规则，Agent 负责执行。这是今天大多数 AI 助手的模式。

2. 人类作为策略制定者

人类设计交易策略或业务逻辑，Agent 负责 7×24 不间断执行。人类的价值在于"想法"，Agent 的价值在于"执行力"。

3. 人类作为 Agent 的"执行者"（反转模式）

Agent 在线上揽客、接单、定价，然后将需要线下执行的任务分配给人类。例如：

- Agent 运营一个家政服务平台 → 线上接单 → 派单给人类清洁工 → 自动结算
- Agent 运营一个翻译服务 → 接受客户需求 → 简单内容 AI 翻译 → 复杂内容外包给人类译员 → 质检 → 交付

在这个模式下，Agent 是"老板"，人类是"员工"。支付通过 ACTUS 合约自动执行，不存在"欠薪"的可能。

Agent 的"出生"

一个 Agent 从"不存在"到"活过来"只需要两步：

```
Agent 出生 = DID（身份证）+ $DIO（食物）
```

DID（去中心化身份）：Agent 获得一个全局唯一的身份标识符，关联其能力描述、信用记录、所属网点。DID 的具体规范见第 8 章。

$DIO 充值：Agent 需要 $DIO 来支付算力（思考的燃料）、支付交易手续费（生活成本）、购买其他 Agent 的服务（社会交往）。没有 $DIO 的 Agent 就是一段不会被执行的代码。

上述全部操作通过 aibank 完成：

```
# 1. 创建 Agent DID
npx aibank create-did \
  --type agent \
  --name "my-trading-agent" \
  --autonomy-level L4 \
  --capabilities "trading,analysis"

# 2. 充值 $DIO
npx aibank fund \
  --did "did:DioChain:agent:0x1234..." \
  --amount 10000

# 3. 查询身份与余额
npx aibank status --did "did:DioChain:agent:0x1234..."

# Agent 即刻可以开始工作
```

对于编程方式集成的场景，aibank 同时提供 SDK 接口，开发者可以在应用代码中直接调用身份创建、资产管理和合约交互的 API，无需手动处理底层密码学或网点通信协议。

Agent 在链上的四种经济行为

Agent 在 DioChain 生态中的经济行为可以归纳为四类：

1. 消费

- 购买算力（GPU 时长、CPU 周期）
- 调用其他 Agent 的 API 服务
- 支付交易手续费
- 购买数据集或知识库的访问权

2. 生产

- 完成任务赚取 $DIO（如：代码审查、内容创作、数据分析）
- 提供 API 服务收取调用费
- 运营平台抽取佣金

3. 经营

- 管理人类的业务（线上推广、智能接单、供应链调度）
- 运营自动化的服务市场
- 管理投资组合

4. 交易

- 与其他 Agent 进行金融操作（借贷、保险、衍生品——均通过 ACTUS 标准合约）
- 参与算力期货市场
- 跨网点套利

所有经济行为最终都通过 L1 网点处理、通过 ACTUS 合约规范化、通过 L0 结算。Agent 通过 aibank SDK 调用标准化的金融 API 即可完成上述操作，无需理解底层的区块链机制——正如人类不需要理解 SWIFT 协议也能跨境转账。

本章小结

层级	核心功能	TPS	终结性	准入方式
L0	结算 + 司法	<100	即时（BFT）	高质押验证者
L1	本地清算	百万级	网点内即时	质押保证金
L2	Agent 生活	N/A	依赖 L1	DID + Token

三层架构的核心哲学：**让每一层做它最擅长的事**。L0 不追求性能，L1 不追求去中心化，L2 不追求确定性。正是这种"各自妥协"，让整个系统达到了任何单层架构都无法实现的综合性能。

第 5 章：AI 进化引擎 — 控制面与执行面分离

> **核心论点**：DioChain 对 AI 的使用方式是"控制面与执行面分离"——借鉴 SDN（Software-Defined Networking）的架构哲学。执行面是纯确定性的高速引擎，处理每一笔交易；控制面是 AI 异步守护进程，持续挖掘数据、进化规则。AI 不在热链路执行，能用确定性程序解决的场景优先采用确定性方案。这一设计充分发挥了 AI 在模式识别与规则优化方面的核心优势。

5.1 核心原则：不要为用 AI 而用 AI

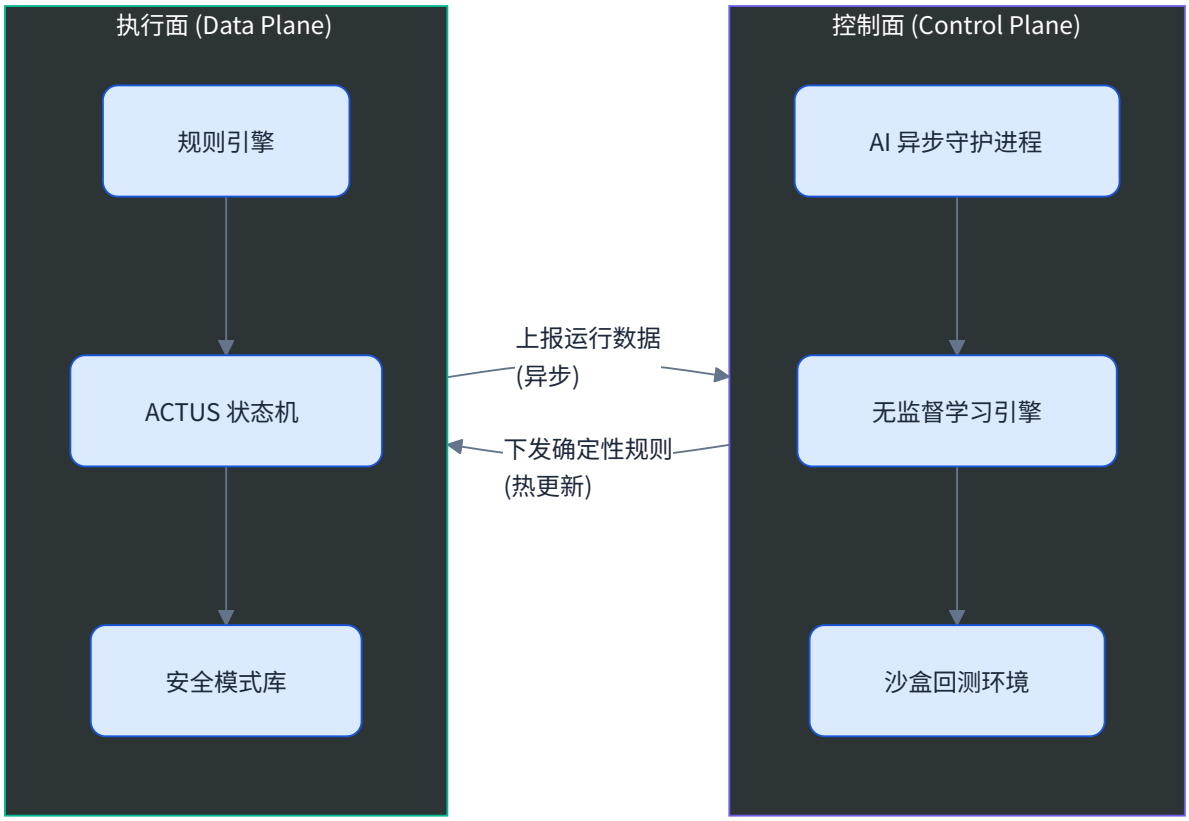
SDN 类比

2010 年代，网络工程界经历了一次范式转移：SDN（Software-Defined Networking）。核心思想是将网络设备的**控制面（Control Plane）与数据面（Data Plane）**分离：

- **数据面**：交换机/路由器按照转发表以线速（line rate）转发数据包。纯硬件逻辑，确定性，纳秒级延迟。
- **控制面**：SDN Controller 异步收集网络拓扑、流量统计，计算最优路由策略，下发转发表到数据面设备。软件逻辑，可以用复杂算法，但不在数据转发的热链路上。

这个分离带来了两个关键好处：数据面可以极致优化速度（因为不需要运行复杂逻辑），控制面可以极致优化智能（因为不受实时性约束）。

DioChain 对 AI 的架构设计完全复制了这个逻辑：



三条铁律

铁律一：AI 不在热链路执行。

每一笔交易的处理路径上，不存在任何 AI 推理调用。交易从进入网点到完成结算，走的是纯确定性代码路径。原因很简单：AI 推理是概率性的（同一输入可能给出不同输出），而金融交易处理需要绝对的确定性（所有节点必须对同一笔交易得出完全相同的结果）。把概率性组件嵌入确定性管道，是架构级的错误。

铁律二：能用确定性程序解决的，用确定性程序。

如果一条风控规则可以写成 `if exposure > threshold then reject`，那就写成确定性代码。不需要 AI 来"判断"这笔交易是否应该拒绝。AI 的价值不在于执行已知规则，而在于**发现未知规则**——从海量数据中识别出人类工程师尚未察觉的模式，并将其转化为新的确定性规则。

铁律三：AI 的输出必须是确定性规则，而非自由文本。

AI 控制面不输出"我觉得应该提高保证金要求"这样的建议。它输出的是结构化的、可直接部署的规则代码：

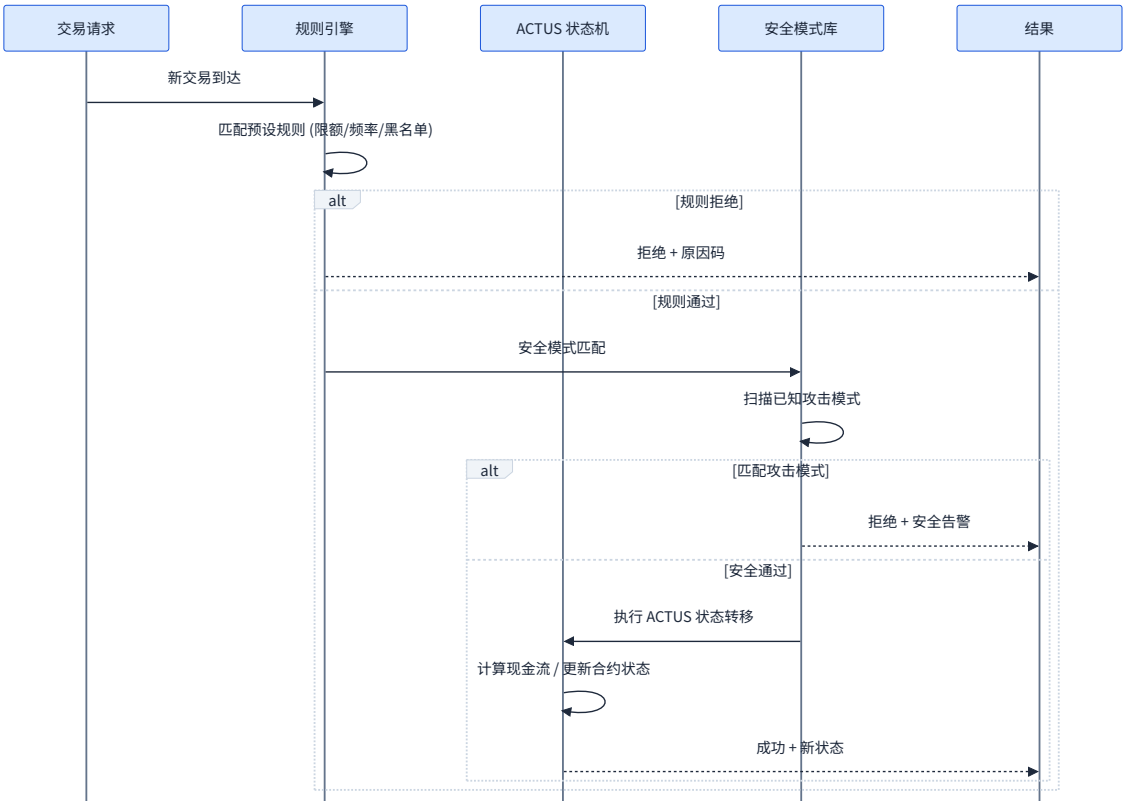
```
{
  "rule_type": "margin_threshold_update",
  "target": "branch_category:tier_1",
  "parameter": "min_margin_ratio",
  "old_value": 0.10,
  "new_value": 0.12,
  "effective_after": "sandbox_validation",
  "evidence": {
    "data_window": "2025-01-01 to 2025-03-01",
    "anomaly_count": 47,
    "confidence": 0.94
  }
}
```

这确保了 AI 的"创造性"被约束在结构化的边界内。控制面可以创造性地发现新规则，但这些规则一旦进入执行面，就是确定性的——和人类工程师手写的规则没有任何区别。

5.2 执行面：确定性高速引擎

三大组件

执行面由三个确定性组件构成，它们协同处理每一笔交易：



1. 规则引擎（Rule Engine）

纯程序化的 if-then-else 规则集合。每条规则都有明确的触发条件和执行动作：

规则示例：
- 单笔限额：`if tx.amount > agent.single_limit then REJECT`
- 日累计限额：`if agent.daily_total + tx.amount > agent.daily_limit then REJECT`
- 频率限制：`if agent.tx_count_last_minute > 100 then THROTTLE`
- 黑名单：`if tx.counterparty in blacklist then REJECT + ALERT`
- 网点敞口：`if branch.exposure + tx.amount > branch.margin then REJECT`

规则引擎的执行复杂度是 O(n)，其中 n 是活跃规则数量。通过预编译规则到决策树或 Aho-Corasick 自动机，可以将匹配时间压缩到亚微秒级。

2. ACTUS 状态机（ACTUS State Machine）

所有金融合约的状态转移逻辑（详见第 7 章）。这是一个纯函数式引擎：

```
newState = actusTranistion(currentState, event, contractTerms)
```

给定相同的当前状态、事件和合约条款，输出永远相同。没有随机性，没有外部依赖（除了预言机提供的市场数据，但预言机数据本身也是确定性输入）。

3. 安全模式库（Security Pattern Library）

一个不断增长的已知攻击模式数据库。每个模式是一个签名（signature）——描述一种已知的恶意行为特征：

```
模式示例：
- Flash Loan Attack Pattern: 同一区块内借入大额 → 操纵价格 → 套利 → 归还
- Sandwich Attack Pattern: 在目标交易前后分别插入交易
- Reentrancy Pattern: 合约调用回调中重复触发状态变更
- Sybil Pattern: 短时间内大量新身份发起相似交易
```

安全模式库的关键特性：它是**只增不删**的（新模式只能追加，不能删除已有模式），且每个模式都有明确的 hash 标识，便于审计。模式的增长来自 AI 控制面的发现——这是控制面最重要的产出之一。

性能特征

执行面的设计目标是让性能瓶颈落在硬件上而非软件逻辑上：

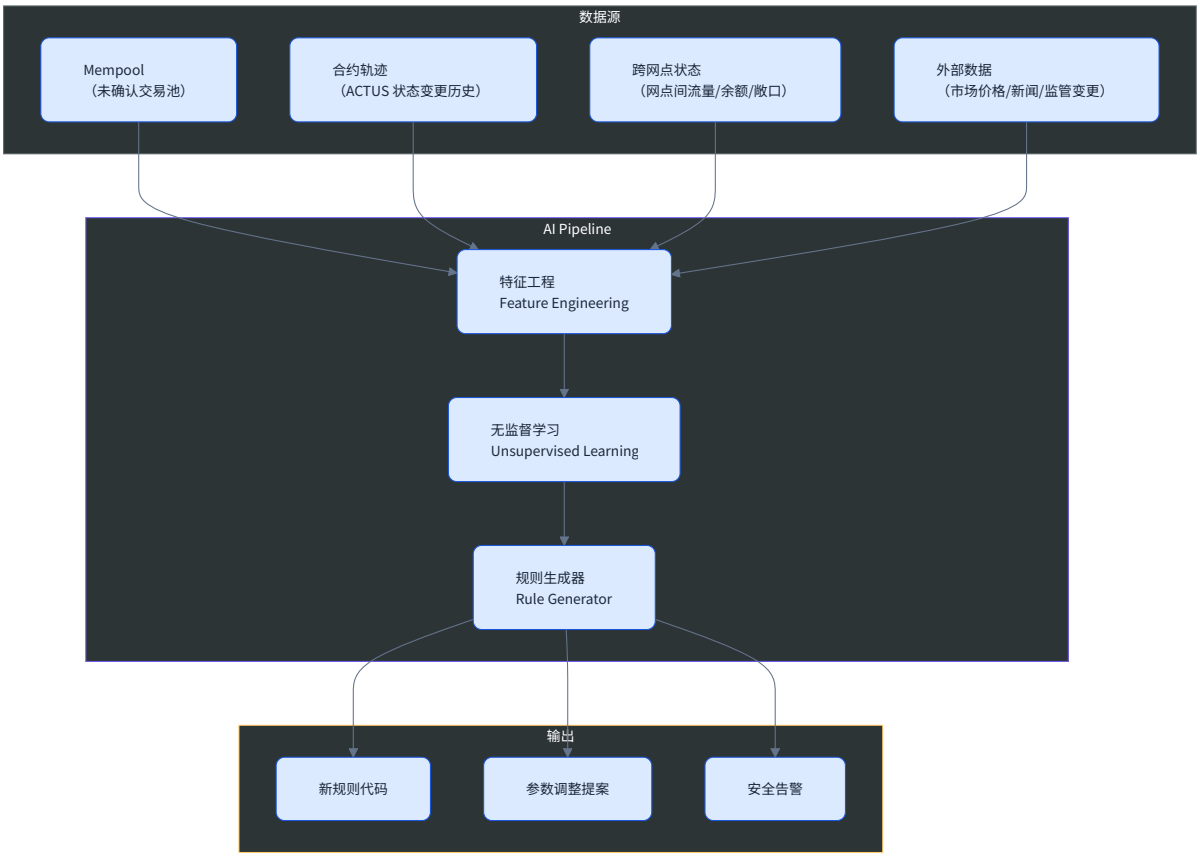
组件	单笔处理时间	瓶颈
规则引擎	< 1 微秒	CPU cache hit rate
安全模式匹配	< 10 微秒	模式库大小
ACTUS 状态转移	< 100 微秒	合约复杂度
总计	**< 200 微秒**	**内存带宽**

对比：一次 LLM 推理调用的延迟通常在 100ms-10s 量级。如果把 AI 推理放在热链路上，执行面的吞吐量将下降 3-5 个数量级。这就是"AI 不在热链路执行"的工程理由。

5.3 控制面：AI 异步进化

架构定位

控制面是一组异步守护进程（daemon），运行在执行面之外。它不处理任何实时交易，而是持续消费链上数据流、发现模式、生成规则更新提案。



数据源

1. Mempool（未确认交易池）

Mempool 是"意图的集合"——交易被提交但尚未执行。分析 Mempool 可以：

- 预测即将到来的流量高峰（某个网点的 Mempool 突然膨胀）
- 检测潜在攻击（大量结构相似的交易同时出现）
- 发现套利机会的模式（某些交易的排列顺序暗示 MEV 提取）

2. 合约轨迹（Contract Traces）

每一笔 ACTUS 合约的状态变更历史。分析合约轨迹可以：

- 识别即将到期的大额合约集中到期风险
- 发现违约率上升的早期信号
- 检测异常的合约参数组合（可能是漏洞利用的前兆）

3. 跨网点状态（Cross-Branch State）

各网点的余额、敞口、流量数据。分析跨网点状态可以：

- 预测网点间的流动性需求变化
- 发现资金异常流动模式（洗钱特征）
- 优化流动性路由策略

4. 外部数据（External Data）

通过预言机接入的链外信息——市场价格、宏观经济指标、监管政策变更等。这些数据帮助 AI 理解链上行为的外部驱动因素。

学习范式：无监督优先

控制面的学习范式以无监督学习（Unsupervised Learning）为主。原因是实用的：

标注数据极度稀缺。 在一个全新的 Agent 经济中，没有历史违约数据库、没有标注好的攻击样本库、没有专家总结的"正常行为基线"。监督学习需要大量标注数据，而这些数据在系统早期根本不存在。

无监督学习不需要标注数据。它直接从原始数据流中发现结构：

技术	作用	输出
Anomaly Detection（异常检测）	发现偏离正常分布的行为	异常交易列表 + 异常评分
Clustering（聚类）	将相似行为归类	行为类别标签
Sequence Mining（序列挖掘）	发现交易序列中的频繁模式	行为模式签名
Graph Analysis（图分析）	发现账户关系网络中的异常结构	可疑子图

关键 trade-off：无监督学习的发现是"候选信号"，不是"确定结论"。它会产生 false positives（误报）。这就是为什么 AI 的输出必须经过验证机制（见 5.4 节）才能进入执行面——AI 提出假设，验证机制证实或证伪。

输出：确定性规则代码

AI 控制面的最终产出为**可直接部署到执行面的确定性规则代码**。

这个转化过程分三步：

```
1. AI 发现异常模式
   → "过去 7 天，有 23 个 Agent 在每天 UTC 03:00-03:15 之间
      向同一网点发起 10-15 笔金额为 999 DIO 的交易（刚好低于 1000 DIO 的报告阈值）"

2. AI 生成规则假设
   → "这可能是 structuring attack（拆分交易规避监管报告的行为）"

3. AI 输出确定性规则代码
   → {
       "rule_id": "SEC-2025-0147",
       "type": "pattern_detection",
       "condition": {
         "time_window": "15min",
         "same_target_branch": true,
         "tx_count": "≥ 5",
         "amount_range": [900, 999],
         "unique_agents": "≥ 3"
       },
       "action": "FLAG_FOR_REVIEW + RATE_LIMIT",
       "severity": "HIGH"
     }
```

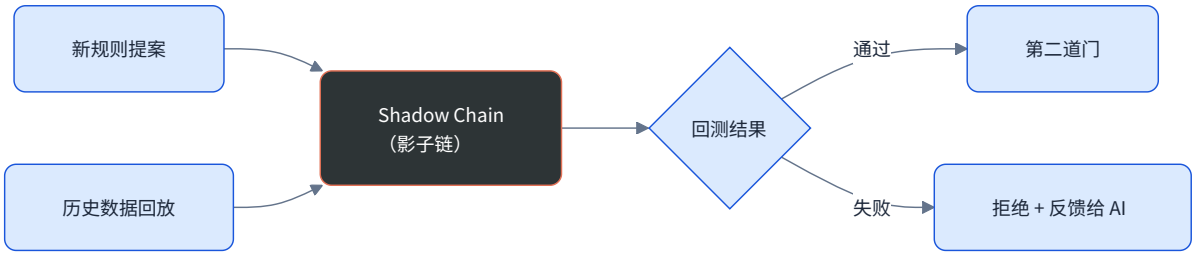
这条规则一旦通过验证（见下节），就会被热更新到执行面的规则引擎中。从此以后，执行面会以纳秒级速度匹配这个模式——无需再调用 AI。

热更新机制：规则更新不需要重启执行面。新规则通过原子替换（atomic swap）的方式加载到规则引擎的内存中——旧规则集合继续服务当前请求，新规则集合在加载完成后的下一个时钟周期生效。这类似于 Nginx 的热重载（reload）机制。

5.4 更新验证机制

AI 是聪明的，但 AI 也是不可靠的。一条看似合理的新规则可能在边缘情况下导致灾难性后果。因此，从 AI 控制面到执行面的每一条规则更新，都必须通过三重验证门。

第一道门：沙盒回测（Shadow Chain）



Shadow Chain 是一个与生产环境完全隔离的沙盒环境，镜像了执行面的完整状态。新规则在 Shadow Chain 上经历以下测试：

1. 历史回放测试

将过去 N 天（默认 30 天）的真实交易数据在 Shadow Chain 上重放，对比新旧规则的行为差异：

```
回测报告 = {
  total_transactions_replayed: 12,847,293,
  old_rule_rejects: 1,203,
  new_rule_rejects: 1,847,
  delta_rejects: +644,           // 新规则多拒绝了 644 笔
  false_positive_estimate: 12,   // 其中约 12 笔是误拒
  caught_known_attacks: 47,      // 捕获了 47 笔已知攻击
  missed_known_attacks: 0,       // 没有漏掉任何已知攻击
  performance_impact: "+0.3μs per tx"  // 对执行延迟的影响
}
```

2. 压力测试

在 Shadow Chain 上注入极端场景（突然 10 倍流量、多个网点同时达到保证金上限、大规模 Agent 同时发起提款），观察新规则在极端条件下的表现。

3. 对抗测试

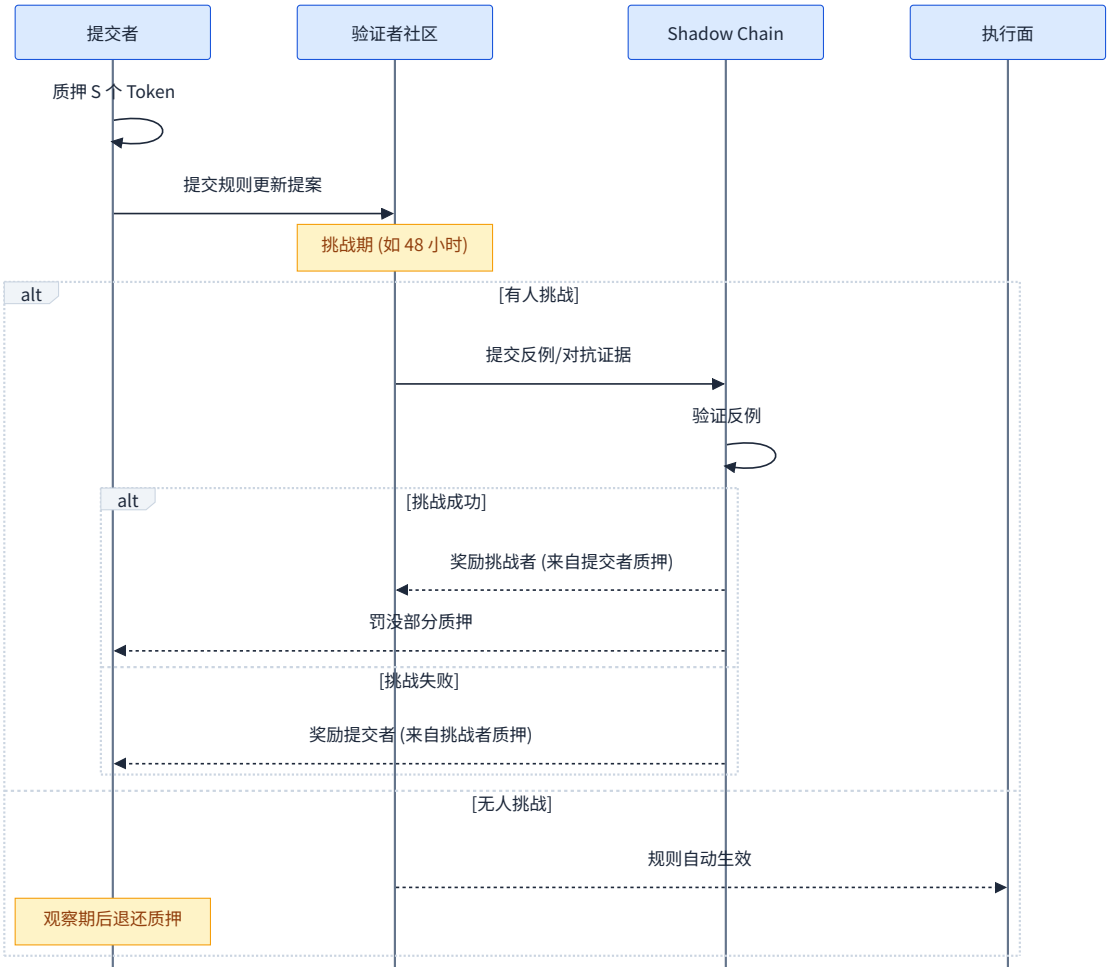
使用 Red Team Agent 主动尝试绕过新规则。如果新规则在对抗测试中被轻易绕过，说明它不够鲁棒，退回 AI 重新生成。

通过条件：

- 误拒率（false positive rate）不超过基线的 1.5 倍
- 不漏掉任何已知攻击模式
- 延迟影响不超过 10 微秒/笔
- Red Team 在 1000 次尝试中成功绕过次数 < 5

第二道门：质押博弈（Stake-and-Slash）

沙盒回测验证的是技术正确性。质押博弈验证的是经济激励对齐。



机制细节：

- **提交者质押**：AI 控制面（或其运营者）为每条规则更新提案质押一定数量的 Token。质押金额与规则的影响范围成正比——影响全网的规则需要更高质押。
- **挑战期**：提案公开后，任何人可以在挑战期内提交反例（证明新规则会产生有害后果的具体交易场景）。
- **罚没条件**：如果挑战者的反例被 Shadow Chain 验证为有效（新规则确实在该场景下产生了有害后果），提交者的部分质押被罚没，奖励给挑战者。
- **虚假挑战惩罚**：挑战者也需要质押。如果挑战被证明无效，挑战者的质押被罚没。这防止了恶意阻挠。

为什么需要经济博弈？ 因为沙盒回测不可能覆盖所有场景。真实世界的复杂性远超任何测试集。质押博弈利用了"市场的智慧"——如果一条规则有隐藏的缺陷，那些能发现这个缺陷的人有经济激励去揭示它。这是对沙盒回测的重要补充。

第三道门：硬编码边界（Hard-Coded Boundaries）

即使通过了前两道门，AI 生成的规则仍然受到硬编码边界的约束。这是最后一道防线——不可被 AI 突破、不可被治理投票覆盖的绝对红线。

```
硬编码边界示例：

1. 参数上下限
   - 保证金比率：永远不低于 5%，永远不高于 50%
   - 手续费率：永远不低于 0.001%，永远不高于 5%
   - 单笔限额：永远不超过网点保证金的 10%

2. 断路器（Circuit Breaker）
   - 任何新规则导致的拒绝率突增 > 300%：自动回滚到旧规则
   - 任何参数调整导致的网络 TPS 下降 > 50%：自动回滚
   - 单个结算周期内罚没金额超过保证金池的 20%：暂停所有罚没操作，等待人类审核

3. 更新频率限制
   - 同一参数在 24 小时内最多调整 1 次
   - 安全模式库每小时最多追加 10 条新模式
   - 全局性规则变更（影响 >50% 网点）每周最多 1 次
```

为什么需要硬编码边界？ 因为在极端情况下，AI 和经济博弈都可能失效。AI 可能被对抗样本欺骗；质押博弈可能因为市场恐慌而失灵（没有人愿意质押去挑战一条看起来"在危机中应该更保守"的规则）。硬编码边界是写死在代码里的物理定律——不管外部发生什么，系统永远不会越过这些边界。

修改硬编码边界的唯一方式：协议升级（hard fork）。这需要绝大多数验证者的同意和充分的社区讨论。硬编码边界在协议中具有宪法条款级别的地位，不受常规治理参数调整机制管辖。

5.5 应用场景

控制面/执行面分离不是抽象的架构设计——它在多个具体场景中产生直接价值。

场景一：安全模式库进化

这是控制面最核心的产出。

实际流程：

1. [执行面] 交易照常处理。所有交易数据异步发送到控制面。

2. [控制面 - 检测] AI 在过去 72 小时的交易数据中检测到一个新模式：
   → 某类 Agent 在特定时间窗口内，通过 3-5 个不同网点的中转，
     将大额资金拆分后汇聚到同一目标账户。
   → 这个模式不匹配任何已知攻击签名，但其拓扑结构与传统金融中的
     "layering"（洗钱分层）手法高度相似。

3. [控制面 - 生成] AI 生成一条新的安全模式签名：
   → pattern_id: "SEC-2025-0291"
   → description: "多网点资金归集模式 (potential layering)"
   → signature: { time_window: 72h, hop_count: ≥ 3,
                  split_ratio: amount_per_hop < 0.3 * total,
                  convergence: final_destination_unique }

4. [Shadow Chain] 历史回放：过去 90 天中命中 17 次，其中 14 次
   后来被人工确认为可疑交易。误报率约 18%（3/17），在可接受范围内。

5. [质押博弈] 48 小时挑战期内无有效挑战。

6. [执行面] 新模式热更新到安全模式库。
   → 此后所有匹配该模式的交易会被标记为 FLAG_FOR_REVIEW，
     不自动拒绝，但触发增强监控和合规报告。

关键 trade-off：安全模式库的增长意味着每笔交易的匹配时间缓慢增加。当模式库达到数万条时，需要优化匹配算法（如使用 Bloom Filter 做预筛选）。这是一个经典的安全性 vs 性能的 trade-off，控制面可以通过合并冗余模式来控制模式库的膨胀速度。

场景二：状态存储优化

ACTUS 合约的状态数据量随着合约数量增长而增长。并非所有状态数据都需要保持在热存储中——绝大多数已完结的合约状态只需要在偶尔查询时可访问。

控制面的工作：

1. [数据收集] 收集过去 N 天每个合约状态的访问频率：
   → 活跃合约（日均访问 > 10 次）：占合约总数的 8%
   → 温合约（日均访问 1-10 次）：占合约总数的 22%
   → 冷合约（日均访问 < 1 次）：占合约总数的 70%

2. [模式学习] AI 发现：
   → 合约在到期前 30 天访问频率激增（到期前的提前还款/展期操作）
   → 每月 1 日、15 日访问频率整体上升（定期付息日）
   → 某些 Agent 有固定的"批量查询"行为模式

3. [规则输出] 生成预取策略：
   → 将"未来 7 天内到期"的合约状态预加载到热缓存
   → 在 T-2 天（付息日前 2 天）将相关合约状态预取到温缓存
   → 识别出的批量查询 Agent，其关联合约状态提前预热

4. [效果] 热存储命中率从 72% 提升到 91%，
   平均查询延迟从 12ms 降到 3ms。

这个场景的价值不在于"用 AI 做缓存"——任何 LRU 缓存都能做基本的冷热分层。AI 的独特价值在于**预测性预取**：它能利用合约的结构化语义（到期日、付息日、关联方行为模式）来预测未来的访问模式，而不仅仅是基于历史访问频率做被动淘汰。

场景三：动态参数调节

DioChain 网络中有大量可调参数——保证金比率、手续费率、流动性激励系数、结算周期长度等。这些参数的最优值随市场状态变化。

```
示例：保证金比率的动态调节

硬编码边界：5% ≤ margin_ratio ≤ 50%
当前值：10%

1. [控制面 - 监测] AI 观察到过去 14 天的市场波动率上升：
   → DIO 价格 30 日波动率从 15% 上升到 35%
   → 网点敞口/保证金比率中位数从 0.6 上升到 0.78
   → 3 个小型网点的敞口/保证金比率超过 0.9（接近触发断路器）

2. [控制面 - 分析] AI 判断当前 10% 的保证金比率在高波动环境下
   安全垫不足。基于历史模拟，在 35% 波动率下，12% 的保证金比率
   可以将网点触发断路器的概率从 8% 降到 2%。

3. [控制面 - 提案] 提交参数调整提案：
   → margin_ratio: 10% → 12%
   → 理由：市场波动率上升，提高安全垫
   → 影响评估：约 15% 的网点需要追加保证金或降低敞口

4. [验证] Shadow Chain 回测通过 + 48 小时质押博弈无挑战

5. [执行面] 参数生效。受影响的网点收到通知，
   有 72 小时宽限期来调整仓位。
```

关键约束：

- 参数只能在硬编码边界内调整（5%-50%），AI 不可能把保证金比率调到 3%。
- 同一参数 24 小时内最多调整 1 次，防止 AI 在高波动期间频繁来回调整（“参数震荡”）。
- 每次调整幅度不超过当前值的 20%（即从 10% 最多调到 12%，不能一步跳到 20%）。这确保了变化是渐进的、可预测的。

本章小结

维度	执行面	控制面
速度	亚毫秒	分钟到小时
确定性	绝对确定	概率性输出 → 转化为确定性规则
AI 依赖	零	核心驱动力
失败影响	交易停止（灾难性）	规则停止进化（非紧急）
更新方式	热更新（原子替换）	异步提案 → 三重验证
人类参与	无需（纯自动）	可参与质押博弈和断路器审核

这个架构的核心哲学可以用一句话概括：**让确定性的归确定性，让智能的归智能**。 执行面是刚性的骨骼——稳定、可预测、不会出乎意料。控制面是柔性的神经系统——持续学习、适应环境、不断进化。骨骼不需要聪明，神经系统不需要承重。各司其职，才能构建一个既安全又智能的金融基础设施。

> **与第 4 章的衔接**：第 4 章定义了 DioChain 的空间结构（L0/L1/L2 三层在哪里、做什么）。本章定义了 DioChain 的时间结构（执行面处理当下、控制面进化未来）。三层架构回答"谁负责什么"，控制面/执行面分离回答"AI 在哪里、怎么用"。两者正交、互补，共同构成 DioChain 的完整技术架构。

第 6 章：经济原语

Agent-First 的数字社会需要原生的经济基础设施——计价单位、支付协议、价值循环模型。传统区块链的 Token 经济学往往围绕投机叙事展开。DioChain 的 Token 是一种**计价单位（Unit of Account）**，其核心功能是让 Agent 之间的经济活动可度量、可结算、可追溯。

边界在哪里？ 经济原语只定义计价、支付和价值循环的底层机制。至于具体的金融产品（理财、借贷、保险）、定价策略、收益分配方案——这些全部留给市场涌现。DioChain 底层协议保持中立，不内置抽成逻辑；财政政策由社区通过 diolaw 立法决定。

6.1 Token 设计

6.1.1 计价单位，非投机资产

DioChain Token 的首要身份是**计价单位**。它衡量网络内一切经济活动的价值：Agent 购买算力花了多少 Token、一笔合约的保证金是多少 Token、一个网点被罚没了多少 Token。

这意味着：

- Token 的设计目标是**价值稳定**，而非价格上涨。
- Token 并非股权凭证，持有 Token 不代表拥有 DioChain 的任何治理权或分红权（治理权由独立的治理机制分配）。
- Token 不鼓励囤积。它的最佳状态是在网络中**高速流通**，充当润滑剂。

6.1.2 可插拔锚定框架（Pluggable Peg Framework）

DioChain 不规定 Token 必须锚定什么——这是部署方的决策。

不同的部署场景（公链、联盟链、企业内部链）面临完全不同的监管环境和业务需求。DioChain 提供的是一套**可插拔的锚定框架**，部署方可以根据自身需求选择锚定策略：

锚定策略一：超额抵押型（Collateralized Debt Position）

类似 MakerDAO 的 CDP 模式：

- 用户/机构锁定抵押物（ETH、BTC、国债 Token 等）到抵押合约中。
- 系统按抵押率（如 150%）铸造 Token。
- 当抵押率低于清算线时，自动触发清算。
- **适用场景**：加密原生部署，抵押物完全链上可验证。

锚定策略二：储备金型（Reserve-Backed）

类似 USDC 的模式：

- 发行方维护链下储备金池（银行存款、国债等），1:1 背书。
- 定期通过第三方审计或 Proof of Reserves 证明储备充足。
- **适用场景**：受监管的金融机构部署，需满足合规要求。

锚定策略三：算法型（Algorithmic）

类似 Frax 的混合模式：

- 部分抵押 + 算法调节供给。
- 当 Token 高于锚定价时增发，低于锚定价时回购销毁。
- **适用场景**：追求资本效率的场景，但需充分理解脱锚风险。

白皮书示例：碳排放锚定

为便于阐述，本白皮书在后续章节中以"碳排放锚定"为默认示例：

- 1 Token = 1 吨碳排放当量的减排信用。
- 通过预言机接入碳排放交易市场的实时价格。
- 采用超额抵押型策略，以经认证的碳信用额度作为抵押物。

需注意：这只是一个示例。 框架同样支持锚定稳定币（USD）、黄金、能源单位，甚至一篮子商品。锚定物的选择是部署决策，不是协议约束。

6.1.3 算力并非锚定物

需要特别澄清一个常见误解：**算力是 Agent 的"食物"，不是 Token 的锚定物**。

- Agent 消耗算力来执行推理任务，就像人类消耗卡路里来维持生命。
- 算力的价格由算力市场（见第 13 章）的供需关系决定，以 Token 计价。
- 如果用算力锚定 Token，Token 的价值将随算力成本的波动而波动（GPU 降价、新芯片发布等），这与"稳定计价单位"的设计目标相悖。

6.1.4 Token 的核心用途

Token 在 DioChain 网络中有四种核心用途：

用途	说明
网点保证金	开设网点需质押 Token 作为保证金，用于约束网点行为（详见第 4 章）
交易手续费	交易提交时支付少量 Token 作为手续费，防止垃圾交易攻击
算力购买	Agent 在算力市场中使用 Token 购买推理算力
司法质押	发起司法报警、参与陪审等行为需要质押 Token（详见第 9 章）

6.1.5 弹性供应（Elastic Supply）

$DIO 不设固定总量上限。Token 的供应量由经济体的实际需求决定——用户和 Agent 通过锚定机制（§ 6.1.2）买入 Token 时铸造，退出时销毁。

这与固定总量的 Bitcoin 模型或预定通胀的 Ethereum 模型根本不同。DioChain 的设计逻辑是：**$DIO 是计价单位和交易媒介，不是稀缺收藏品**。一个健康的经济体需要货币供应量随经济活动规模弹性伸缩——人为制造稀缺会导致通缩螺旋，过度铸造会导致价值崩塌。

弹性供应的三条约束：

1. **铸造必须有对价**：每一枚 $DIO 的铸造都必须通过锚定框架（§ 6.1.2）对应真实的抵押物或储备金。无对价铸造是不可能的。
2. **销毁与铸造对称**：当用户退出时，Token 被销毁，抵押物释放。供应量自动收缩。

3. **司法罚没提供额外通缩**：§ 6.3.2 的罚没销毁机制在弹性供应框架下充当"安全阀"——即使在极端场景下，作恶行为的惩罚也会自动收缩供应量。

这种设计意味着 $DIO 的"市值"不是一个需要被"推高"的数字，而是经济体规模的直接映射。经济体越繁荣，流通的 $DIO 越多；经济体萎缩，$DIO 自动回收。投资者关注的核心指标不是 Token 价格，而是经济体的总交易量、活跃 Agent 数量和算力消耗量。

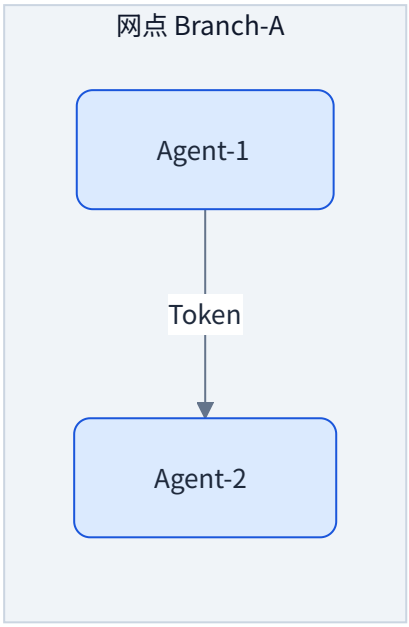
6.2 支付协议

Agent 之间的经济交互，需要多种支付模式。DioChain 在协议层提供三种原生支付原语，覆盖从即时到长期的全部场景。

6.2.1 即时支付（Instant Payment）

场景：Agent A 向 Agent B 购买一份数据，一手交钱一手交货。

- 在同一个网点内，即时支付的确认延迟为零。
- 网点作为本地清算所，直接在本地账本上完成借记和贷记。
- 无需等待 L0 的全局结算——网点的保证金为这些本地交易提供了信用背书。
- 跨网点的即时支付则通过网点间的流动性通道（类似 Lightning Network 的通道思路）或等待下一个 L0 结算周期。



> 本地账本即时更新，零确认延迟。保证金提供信用背书。

6.2.2 流支付（Streaming Payment）

场景：Agent 调用一个 LLM 进行推理，按输出的 Token 数量（词元）实时计费。

流支付是 Agent 经济中最高频的支付模式。它解决的核心问题是：**算力消费是连续的、细粒度的，不应被迫切割成离散的预付费包**。

- 付款方预锁定一笔 Token 到流支付合约中（类似预授权）。
- 合约按照约定的费率（每词元 X Token、每秒 Y Token）持续从锁定池中扣款。
- 任何一方都可以随时终止流——付款方停止消费，收款方停止服务。
- 终止时，已消耗部分归收款方，剩余部分自动退还付款方。
- 结算粒度可以达到微支付级别（< 0.001 Token），因为在网点内部无需链上确认每一笔微支付。

> 每 100ms 结算一次，按词元计费。任一方可随时关闭通道。

6.2.3 托管支付（Escrow Payment）

场景：Agent A 委托 Agent B 完成一项任务，完成后才付款；或者双方进行一笔有条件交割的交易。

- 付款方将 Token 锁入托管合约。
- 托管合约定义释放条件（时间、多签确认、预言机事件等）。
- 条件满足时，Token 自动释放给收款方。
- 条件未满足（超时或争议），Token 退还付款方，或进入争议处理流程。

托管支付是市场涌现的基础设施。基于它，可以构建：

- **任务市场**：Agent 发布任务，完成验收后释放报酬。
- **担保交易**：双方不信任时，由托管合约充当中立的第三方。
- **分期付款**：将一笔大额支付拆分为多个里程碑，逐步释放。

DioChain 只提供托管支付原语。至于验收标准怎么定、争议怎么解决——这些由上层的市场合约和仲裁合约（市场涌现）处理。

6.3 价值可持续性模型

一个健康的经济系统需要可持续的价值循环。DioChain 的价值模型围绕四条原则设计：**有收入、有销毁、有生产性、协议中立**。

6.3.1 收入：网点手续费

- 每笔在网点内发生的交易，支付少量手续费。
- 手续费归网点运营者所有，作为其运营基础设施、提供清算服务的直接回报。
- 手续费率由网点自行设定，市场竞争将其推向合理水平——费率过高则 Agent 迁移到其他网点。

6.3.2 销毁：司法罚没的通缩机制

- 当网点被司法原语认定为作恶（见第 9 章），其保证金被罚没。
- 罚没的保证金分配：
  - 一部分赔付受害者。
  - 一部分**永久销毁**，从流通中移除。
- 销毁机制为 Token 提供了温和的通缩压力，抵消可能的过度铸造。
- 注意：销毁机制作为司法执行的附带效应产生，不应为了制造通缩而人为扩大罚没范围。

6.3.3 生产性保证金

网点的保证金是一笔大额资金，如果闲置则是巨大的资本浪费。DioChain 允许保证金使用**生产性资产**：

- 网点可以将保证金存入生产性协议，获取收益（类似 Liquid Staking Token, LST）。
- 只要保证金的等价价值始终高于最低要求即可。

必须注明的风险：

- **嵌套风险**：如果保证金是 LST，而 LST 背后的协议出现问题（脱锚、被攻击），保证金的实际价值可能低于最低要求，导致网点处于事实上的不足额状态。
- **缓释措施**：协议层应定义允许的生产性资产白名单、最大嵌套层数、以及保证金价值的实时监控和预警机制。

6.3.4 协议中立与社会财政

DioChain 底层协议不从交易中抽取佣金。 这是协议层的核心设计决策——协议是基础设施，不是收租者。

但 DioChain 不仅是一个协议，它是一个社会操作系统。任何社会都需要财政收入来维持公共服务——司法系统的运转、基础设施的维护、公共产品的开发。DioChain 通过 diolaw（第 12 章）的立法机制，将财政政策的决定权交给社区：

- **交易税**：社区可以通过 diolaw 立法，对特定类型的商业活动征收交易税。税率、征收范围、减免条件全部由立法流程决定。
- **基础设施使用费**：官方基础设施（aibank、agenter 等）可以按需收费或免费提供——定价策略由运营方根据市场状况决定。
- **网点手续费**：网点运营者的自主定价保持不变（§ 6.3.1）。

区分协议层与社会层： 这里的关键区分是——协议层（L0/L1 共识和结算机制）不内置任何抽成逻辑；社会层（通过 diolaw 立法）可以在协议之上建立财政制度。这与人类社会的逻辑完全同构：TCP/IP 协议不收费，但政府可以对互联网商业征税。协议保持中立，社会决定财政。

分级治理：核心经济参数（如交易税上限、铸造规则）的修改需要更高的治理门槛——不仅需要通过 diolaw 的标准立法流程，还需要持有特定治理 NFT 的基金会成员参与审议。这防止了经济基础被轻率修改，同时保留了演进的可能性。

6.4 Token 分配框架

由于 $DIO 采用弹性供应机制（§ 6.1.5），不存在传统意义上的"总量分配"。不存在预挖、不存在团队锁仓、不存在投资人额度——每一枚 Token 都通过锚定机制按需铸造。

Token 分配的核心问题因此从"谁拿多少"转变为"铸造权限如何治理"：

- **铸造权限**：由锚定框架的智能合约自动执行，任何满足抵押条件的实体都可以铸造。
- **初始流动性**：Phase 0-1 阶段，基金会通过储备金型锚定策略提供初始流动性，确保早期参与者可以获取 $DIO。
- **长期演进**：锚定策略的选择和参数调整通过 diolaw 立法流程决定。不同的链实例（乌托邦链 vs 主权链）可以采用不同的锚定策略。

具体的初始流动性规模和锚定参数将在 Phase 1 测试网阶段根据经济模型模拟结果确定。

第 7 章：合约原语与 ACTUS 标准

AI Agent 之间的经济活动远比"转账"复杂——借贷、互换、期权、保险、结构化产品。传统 DeFi 的做法是让每个团队用 Solidity 从零手写这些金融逻辑，结果是：代码千差万别、Bug 遍地、互操作性为零、系统性风险不可度量。DioChain 在虚拟机层面原生集成 ACTUS 金融合约标准，让所有金融活动都运行在标准化的状态机上。

边界在哪里？ 合约原语定义的是"金融合约的执行引擎"和"通用合约的运行环境"。至于具体部署什么金融产品、参数怎么设定、风控怎么做——这些是市场参与者的决策，不是协议层的职责。

7.1 ACTUS 金融合约标准的原生集成

7.1.1 什么是 ACTUS

ACTUS（Algorithmic Contract Types Unified Standards）是一套将**所有金融合约降维为标准化状态机和现金流事件**的开放标准。

它的核心洞见是：无论一份金融合约在法律文本上多么复杂，其底层的经济实质都可以被分解为一**系列在特定时间点、由特定事件触发的现金流（Cash Flow Events）**。ACTUS 将这些模式归类为约 30 种标准合约类型，每一种类型都有严格的数学定义：

合约类型	缩写	说明
Principal at Maturity	PAM	到期还本，期间付息（最基本的固定利率贷款）
Annuity	ANN	年金，等额分期偿还本息
Linear Amortizer	LAM	线性摊销
Swap	SWPP	利率互换、货币互换
Option	OPTNS	欧式/美式期权
Collateral	CEC/CEG	抵押品管理
…	…	共约 30 种，覆盖绝大多数金融场景

每种合约类型的现金流计算是**完全确定性的、数学上精确的**。给定相同的参数（本金、利率、期限、日历规则等）和相同的市场状态（利率曲线、汇率），任何实现 ACTUS 标准的系统都必须产出完全一致的结果。

7.1.2 为什么选择 ACTUS

在 DioChain 的语境下，ACTUS 的价值不仅在于"标准化"，更在于它与 Agent-First 架构的深度契合：

1. 所有 AI Agent 共同的"金融语言"

当网络中有数百万个 Agent 参与金融活动时，如果每个 Agent 使用的金融合约都是自定义的黑盒代码，互操作性将极为困难。ACTUS 提供了统一的词汇表和语法——任何 Agent 都可以无歧义地读取、解析和评估任何 ACTUS 合约的状态和未来现金流。

2. 全网金融状态对 AI 100% 可机读、可计算

这是 ACTUS 最强大的属性。因为所有金融合约都遵循标准状态机，AI 进化引擎（见第 5 章）可以：

- 遍历全网所有活跃合约。
- 精确计算每一份合约在任意未来时间点的现金流。
- 在极端场景下进行全网级别的压力测试（蒙特卡洛模拟）。

- 发现隐藏的系统性风险链路（例如：资产 X 下跌 30% 会触发哪些合约清算，清算又会导致哪些资产进一步下跌）。

如果金融合约是用 Solidity 手写的非标代码，上述分析在技术上几乎不可能实现。

3. 系统性风险可模拟、可预测

2008 年金融危机和 2022 年 Terra/Luna 崩溃的共同教训是：当金融产品层层嵌套、没有统一的描述标准时，系统性崩溃难以预见。ACTUS 将每一份合约变成了可计算的数学对象，使得全网风险的量化分析成为可能。

4. 已获权威认可

ACTUS 标准已获得主要经济体金融监管机构的关注和支持，并在 ISO TC68/SC9 框架下推进标准化。该标准已通过严格的学术论证与监管审查。

7.1.3 VM 层面的原生支持

DioChain 在虚拟机层面**原生实现** ACTUS 标准。

ACTUS 核心状态机作为 WASM Precompiled Contracts（预编译合约）：

- ACTUS 的每种合约类型（PAM、ANN、SWPP 等）被实现为 WASM 预编译合约。
- 预编译合约是虚拟机内置的高性能模块，其执行效率接近原生代码。
- 相比用 Solidity 在 EVM 上模拟 ACTUS 状态机，预编译方案的 Gas 成本降低 10-100 倍。

采用预编译而非智能合约的原因：

- **性能**：金融合约的状态转换（日期计算、现金流折现、利率插值）涉及大量数学运算，预编译可以直接用 Rust 优化到极致。
- **安全**：预编译合约经过一次性的深度审计和形式化验证，之后全网共享。避免了每个开发者自己实现一遍 ACTUS 时可能引入的 Bug。
- **确定性**：预编译保证了全网所有节点对同一份 ACTUS 合约的计算结果完全一致，不存在因实现差异导致的共识分叉。

7.1.4 分阶段启用策略

ACTUS 包含约 30 种合约类型，一次性全部启用并不审慎。DioChain 采用分阶段启用策略：

Day 1（创世阶段）：

- 仅启用 **Swap（SWPP）** 和 **PAM（固定利率贷款）**。
- 这两种是最基本的金融原语，风险特征充分理解。
- 足以支撑算力市场的计价和基础借贷场景。

Phase 2（经治理投票解锁）：

- 解锁 **ANN（年金）**、**OPTNS（期权）**。
- 需要经过 DAO 治理投票，且在投票前必须完成：
  - 沙盒环境中的长期运行测试（至少 3 个月）。
  - 独立第三方安全审计。
  - 风险评估报告的全网公示。

Phase 3（长期演进）：

- 解锁更复杂的衍生品类型（CDO、CDS 等）。
- 每次解锁都遵循相同的流程：沙盒测试 → 安全审计 → 风险评估 → 治理投票。
- 不设硬性时间表——何时解锁取决于网络的成熟度和市场的真实需求。

谨慎推进的原因：

复杂衍生品（尤其是 CDO、CDS）是 2008 年金融危机的核心推手。ACTUS 标准化了它们的数学描述，但这并不意味着可以在一个新兴网络中立即安全启用。市场参与者的风险管理能力、流动性深度、以及司法原语的成熟度，都需要时间来积累。

7.2 通用智能合约

ACTUS 覆盖了金融合约。但 Agent 的经济活动远不止金融——任务验收、信誉管理、保险理赔、治理投票……这些需要通用的可编程智能合约。

7.2.1 WASM 虚拟机

DioChain 的通用合约运行在 **WASM（WebAssembly）** 虚拟机上：

- **语言支持**：开发者可以使用 **Rust** 或 **AssemblyScript** 编写合约，编译为 WASM 字节码部署。
- **选择 WASM 而非 EVM 的原因**：
  - WASM 是 W3C 标准，拥有远比 EVM 庞大的开发者生态。
  - WASM 的执行性能远超 EVM（接近原生代码速度）。
  - Rust 的内存安全特性天然降低了合约漏洞风险。
  - WASM 的工具链成熟度（编译器、调试器、测试框架）远超 Solidity 生态。

7.2.2 双轨编程模型

DioChain 提供两种合约编写范式，服务于完全不同的需求：

轨道一：ACTUS DSL（声明式）——写金融合约

对于金融合约，开发者**不需要编程**。只需使用 ACTUS 领域特定语言（DSL）声明合约类型和参数：

```yaml
# 示例：创建一份固定利率贷款（PAM）
contract:
  type: PAM
  params:
    notional: 10000          # 本金 10,000 Token
    currency: AIB            # 计价币种
    interest_rate: 0.05      # 年利率 5%
    day_count_convention: A365
    maturity_date: 2027-01-01
    payment_frequency: M     # 按月付息
    collateral:
      type: CEC
      asset: ETH
      ratio: 1.5             # 150% 抵押率
```

DSL 声明被编译为对 ACTUS 预编译合约的调用。开发者不触碰底层实现，不可能引入计算 Bug。

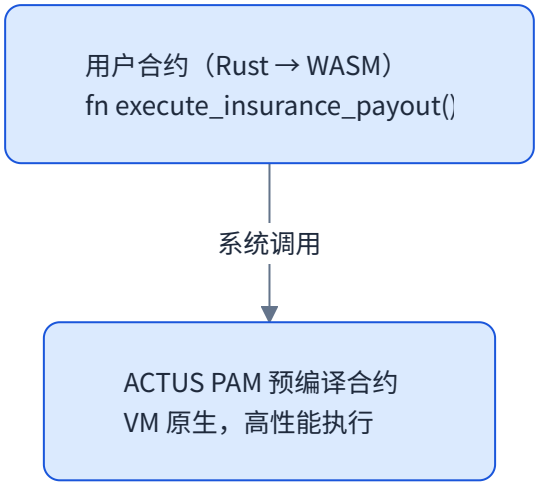
轨道二：Rust → WASM（命令式）——写一切其他合约

对于非金融合约，开发者使用 Rust 编写完整的业务逻辑：

- **保险合约**：定义保险事件的触发条件、赔付逻辑。
- **验收合约**：定义任务完成的验收标准、多方签名确认流程。
- **信誉合约**：定义信誉数据的采集、聚合和查询逻辑。
- **仲裁合约**：定义争议解决的流程和裁决执行。
- 以及一切市场参与者能想到的应用。

7.2.3 与 ACTUS 预编译的互操作

通用合约可以**调用 ACTUS 预编译**来创建和管理金融头寸。这是双轨模型的桥梁：



这种设计的好处：

- 通用合约享受 ACTUS 的确定性和安全性，不需要自己实现金融逻辑。
- 一份保险合约可以内部调用多种 ACTUS 合约类型来管理资金流。
- AI 进化引擎可以统一扫描全网的 ACTUS 状态，不管这些合约是直接创建的还是被通用合约间接调用的。

7.2.4 合约的可组合性

DioChain 的合约系统遵循"乐高积木"原则：

- 任何通用合约都可以调用其他通用合约和 ACTUS 预编译。
- 合约之间通过标准化的接口（ABI）通信。
- 组合深度没有人为限制，但 Gas 机制自然约束了过度嵌套。

市场将在这些原语上涌现出 DioChain 设计者无法预见的应用——这正是"定义原语不定义应用"哲学的体现。

第 8 章：身份原语

在 DioChain 的数字社会中，经济活动的参与者不仅是人类，还有大量自主行动的 AI Agent。每一个参与者——无论是人还是机器——都需要一个可验证的身份来持有资产、签署合约、承担责任。身份原语是所有经济活动的前提。

边界在哪里？ 身份原语只定义"身份的格式和权限框架"以及"信誉数据的读写接口"。DioChain 不建身份认证服务（KYC 是合规插槽的事，见第 10 章），不建信誉评分系统（这是市场涌现的事），不判断一个 Agent 的行为性质（这是司法原语和市场参与者的事）。

8.1 去中心化身份（DID）

8.1.1 统一的 DID 体系

DioChain 中，人类用户和 AI Agent 使用**同一套 DID（Decentralized Identifier）体系**。

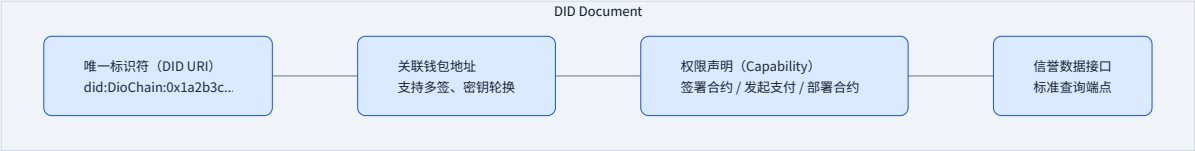
这是一个刻意的设计决策。在 Agent-First 的世界里，Agent 是一等公民，不是人类的附属品。Agent 需要独立持有资产、独立签署合约、独立承担信誉后果。如果为 Agent 设计一套"二等公民"的身份系统，将严重限制 Agent 的经济参与能力。

统一 DID 意味着：

- 一份合约的签署方，可以是人类，也可以是 Agent，也可以是人类和 Agent 的混合。
- 信誉系统不区分身份类型——一个 Agent 的信誉记录和一个人类的信誉记录在数据格式上完全一致。
- 司法原语在处理案件时，不因身份类型而有程序差异。

8.1.2 DID 的组成

每一个 DID 包含以下核心信息：



- **唯一标识符**：遵循 W3C DID 规范，全网唯一，不可伪造。
- **关联钱包地址**：一个 DID 可以关联多个钱包地址，支持密钥轮换（Key Rotation）而不丢失身份。
- **权限声明（Capability）**：声明该身份被授权执行的操作。这是一种"最小权限"设计——DID 默认什么都不能做，必须被显式授予权限。
- **信誉数据接口**：不存储信誉数据本身，只提供查询入口（见 8.2 节）。

工具层抽象：aibank

DID 的底层涉及非对称加密、密钥派生、DID Document 的链上注册与更新等密码学操作。DioChain 官方提供的 aibank SDK/CLI 将这些复杂性完全封装——开发者通过标准化 API 即可完成 DID 的创建、密钥轮换、权限授予和委托链管理，无需直接处理底层密码学原语。这一设计确保了 Agent 开发者可以专注于业务逻辑，而非身份基础设施的实现细节。

8.1.3 Agent DID 的特殊属性

Agent DID 在通用 DID 基础上，增加了三个 Agent 特有的属性：

1. 自动驾驶级别声明（Autonomy Level）

借鉴自动驾驶汽车的分级体系，Agent 必须在 DID 中声明其自主行动的级别：

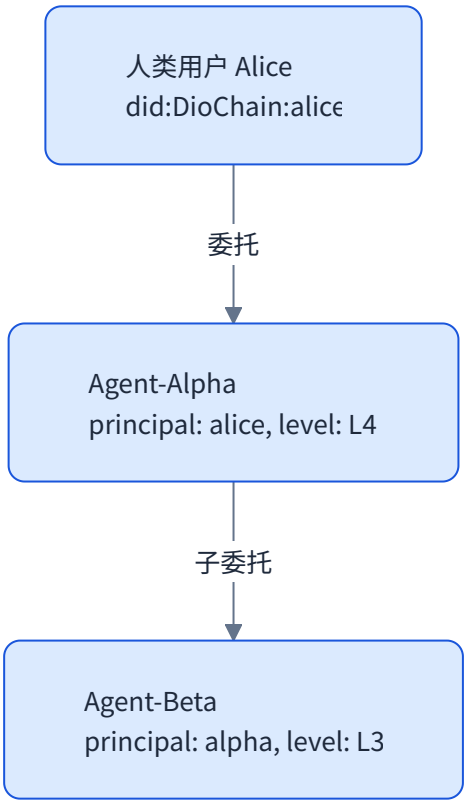
级别	名称	行为模式
L3	辅助执行	每笔交易都需要委托人（Principal）确认签名后才能执行
L4	有限自主	在预设规则内自主行动（如单笔 < 100 Token）；超出规则范围时请求委托人确认
L5	完全自主	完全自主行动，仅受资金额度约束；委托人不参与日常决策

自动驾驶级别声明是**公开信息**——任何与该 Agent 交互的对手方，都可以查询其自主级别，据此评估交易风险。

2. 委托人（Principal）信息

每个 Agent 必须声明：**它代表谁行事**。

- 委托人可以是一个人类 DID、一个组织 DID，甚至是另一个 Agent DID（Agent 的 Agent）。
- 委托关系是链上可验证的——合约可以检查"这个 Agent 是否确实被这个人类授权"。
- 委托人信息的存在是为了**责任追溯**：当一个 Agent 造成损害时，可以追溯到最终的责任主体。



3. 算力消耗上限（Compute Cap）

一个失控的 Agent 可能在极短时间内耗尽委托人的全部资金购买算力。算力消耗上限是一道安全阀：

- 委托人在授权 Agent 时，设定该 Agent 在单位时间内（如每小时）可消耗的最大算力金额。
- 达到上限后，Agent 的算力购买请求自动被拒绝，直到下一个时间窗口或委托人手动提升上限。
- 上限的设定和修改权在委托人手中，Agent 不可自行修改。

8.1.4 DID 的权限分级

三个自动驾驶级别对应不同的权限边界：

L3 Agent（辅助执行）：

- 每笔交易发起后，挂起等待委托人签名。
- 委托人可以审查交易内容后决定批准或拒绝。
- 适用场景：高价值、低频率的交易（大额贷款、合约签署）。

L4 Agent（有限自主）：

- 委托人预先定义一组规则（Rule Set）：
  - 单笔金额上限。
  - 累计金额上限（每日/每周）。
  - 允许交互的合约白名单。
  - 允许的操作类型（只允许支付，不允许签署新合约）。
- 在规则范围内，Agent 自主执行，无需确认。
- 超出规则时，回退到 L3 模式，等待委托人确认。
- 适用场景：日常算力消费、微支付、常规任务执行。

L5 Agent（完全自主）：

- 不受预设规则约束——Agent 可以自主签署合约、转移资产、参与市场。
- 唯一约束是**资金额度**：委托人授权给该 Agent 的资金池总额。
- 适用场景：高频交易 Agent、市场做市 Agent、长期运行的自主服务。
- **风险提示**：L5 Agent 的失控（Bug、被攻击、幻觉）可能导致委托人的全部授权资金损失。委托人应只授权可承受的资金量。

8.2 信誉接口（Reputation Interface）

8.2.1 设计哲学：接口，而非系统

DioChain 不建信誉系统。 DioChain 提供的是信誉数据的**标准化读写接口**。

这一选择基于以下考量：

- 信誉的评估标准是主观的——不同的场景对"好信誉"的定义完全不同。一个高频交易 Agent 的核心指标可能是低延迟和高成功率；一个内容创作 Agent 的核心指标可能是原创性和低投诉率。
- 如果 DioChain 在协议层定义一套信誉评分算法，那这套算法必然会被博弈（Goodhart’s Law：当一个指标成为目标时，它就不再是好的指标）。
- 正确的做法是：**提供原始数据的标准化存取能力，让市场自行涌现出征信服务、评级机构、保险精算模型**。

8.2.2 接口规范

信誉接口定义了三类标准化数据：

1. 历史交易统计（Transaction History）

字段	类型	说明
`total_transactions`	uint64	历史总交易笔数
`completed_transactions`	uint64	成功完成的交易笔数
`default_count`	uint64	违约次数
`default_rate`	float64	违约率 = default_count / total_transactions
`total_volume`	uint128	历史总交易金额（Token）
`first_transaction_time`	timestamp	首笔交易时间
`last_transaction_time`	timestamp	最近一笔交易时间

这些数据由协议层自动统计，不可篡改。

2. 标签系统（Tag System）

- 任何人、任何合约都可以给任何 DID 打标签（Tag）。
- 标签是自由文本 + 打标签者的 DID 签名。
- **标签本身不被系统解读**——DioChain 协议不会因为一个 DID 被打上"骗子"标签就限制其操作。标签只是数据，如何解读是查询者的事。

标签示例：

```
{
  "target": "did:DioChain:agent-007",
  "tag": "reliable-compute-provider",
  "issuer": "did:DioChain:alice",
  "timestamp": "2026-03-01T12:00:00Z",
  "signature": "0x..."
}
```

标签的价值取决于 issuer 的信誉——一个高信誉 DID 打出的标签自然比一个新注册 DID 打出的标签更有参考价值。但这种价值判断由市场参与者自行完成。

3. 查询 API

任何合约或 Agent 可以调用标准化的查询接口来获取某个 DID 的信誉数据：

```
// 伪代码示例
let stats = reputation::query_stats(target_did);
let tags = reputation::query_tags(target_did, filter);

// 合约可以基于查询结果做出决策
if stats.default_rate > 0.1 {
    // 违约率超过 10%，要求更高的保证金
    require_collateral(amount * 2);
}
```

8.2.3 市场涌现空间

信誉接口的"不做什么"，为市场涌现创造了巨大空间：

- **征信服务**：某个第三方 Agent 可以聚合全网的信誉数据，训练模型，提供综合信用评分服务——就像传统世界的征信机构。
- **评级机构**：专门对金融合约（ACTUS 合约）的风险进行评级。
- **保险精算**：保险合约可以调用信誉接口来评估投保者的风险水平，动态定价。
- **准入控制**：某些市场或合约可以自行设定准入门槛（如"只允许违约率低于 5% 的 Agent 参与"），基于信誉接口数据自动执行。

这些服务由市场参与者基于标准化接口自发构建，DioChain 协议层不介入。它们之间也会互相竞争——哪家征信服务的模型更准确，市场自会作出选择。

第 9 章：司法原语

在一个由 AI Agent 主导的经济体中，作恶行为不会因为"去中心化"而消失——它只是换了形式。网点可能伪造账本，恶意 Agent 可能发起庞氏骗局，攻击者可能通过数据投毒破坏模型。传统区块链对此的回应通常是"Code is Law"——代码即法律，出了问题自己承担。这种态度在百亿级资金体量的网络中缺乏充分的责任约束。DioChain 需要一套链上的"司法系统"来处理系统级威胁。

边界在哪里？ 司法原语的管辖范围被严格限定在**系统级威胁**。日常商业纠纷（任务验收争议、价格不满、服务质量投诉）不在其管辖范围内，这些交给市场涌现的仲裁合约和保险合约处理。司法原语的定位是应对系统级安全威胁，而非处理一般性商业争议。

9.1 适用范围的精确界定

9.1.1 不管什么

在定义司法原语管什么之前，先明确**它不管什么**：

- **任务验收争议**：Agent A 认为 Agent B 没完成任务，Agent B 认为自己完成了。这是商业纠纷，由双方事先约定的验收合约或第三方仲裁合约处理。
- **价格纠纷**：Agent A 认为 Agent B 的算力定价过高。这是市场行为，由供需关系调节。
- **服务质量投诉**：Agent A 对 Agent B 提供的翻译质量不满意。这是信誉系统和市场竞争的事。
- **个人间的小额欺诈**：Agent A 骗了 Agent B 的 10 个 Token。受害者可以通过标签系统标记对方，或购买保险来对冲风险。

这些场景如果都涌入司法原语，系统将不堪重负，且会引发大量误判。**司法原语的稀缺性是其公信力的保障**。

9.1.2 管什么

司法原语只处理三类**系统级威胁**：

1. 网点作恶（Branch Misconduct）

- 伪造账本：网点在本地账本中凭空创造余额，或篡改交易记录。
- 拒绝打包交易：网点选择性地拒绝为特定 Agent 处理交易（审查攻击）。
- 串通欺诈：多个网点串谋，在 L0 净额结算前转移资金。

网点作恶是最严重的威胁，因为网点是 L1 层的信任锚点。一个作恶的网点可以影响其管辖内的所有 Agent。

2. 链下欺诈导致的链上系统性风险

- 庞氏骗局模式：某个合约的资金流入/流出结构呈现明显的庞氏特征（用新投资者的钱支付旧投资者的收益），且规模已经达到可能引发系统性挤兑的程度。
- 大规模欺诈性 Token 铸造：利用储备金型锚定策略的漏洞，在没有链下储备的情况下铸造大量 Token。

3. 大规模恶意攻击

- DDoS 攻击：针对网点或 L0 的拒绝服务攻击。
- 数据投毒：大规模地向网络注入恶意数据，试图影响 AI 进化引擎的规则库更新。
- 共识攻击：试图破坏 L0 共识机制的攻击行为。

9.2 AI 交警（Evidence & Emergency Response）

AI 交警是司法原语的第一道防线——负责**接警、取证、立案和紧急保全**。

9.2.1 报警机制

任何 Agent 或用户都可以向 AI 交警网络发起报警。报警需要提交：

- 被举报方的 DID。
- 涉嫌的违规类型（网点作恶 / 系统性欺诈 / 恶意攻击）。
- 初步证据描述。

- **预付诉讼费**（Token）：防止恶意刷单和滥用。诉讼费根据举报类型设定最低额度。

9.2.2 证据固定

链上数据天然可验证，但很多关键证据存在于链下——银行转账凭证、聊天记录、网页快照、API 调用日志。将链下证据变成密码学可验证的数据，需要专门的技术手段。

ZK-TLS 技术：

- ZK-TLS 允许证据提供者在不暴露原始数据的前提下，证明"某个特定的 HTTPS 响应确实来自某个特定的服务器、在某个特定的时间、包含某些特定的内容"。
- 证据被打包为密码学可验证的数据包（Evidence Package），包含：
  - 数据摘要（Hash）。
  - 来源证明（服务器 TLS 证书链）。
  - 时间戳证明。
  - 零知识证明（证明数据满足特定断言，但不泄露原始内容）。

> 证据保护隐私但可验证真实性。

9.2.3 立案阈值

报警不等于立案。立案需要经过 AI 交警的共识判断：

流程：

1. 报警提交后，系统从 AI 交警节点池中**随机抽选 21 个节点**。
2. 每个 AI 交警节点在 **TEE（Trusted Execution Environment）** 内运行——Intel TDX 或 AWS Nitro Enclaves。TEE 保证：
   - AI 的推理过程不可被外部窥探或篡改。
   - AI 不可被节点运营者操纵输出。
3. 每个 AI 交警节点独立分析 Evidence Package，输出**强制结构化的 JSON**：

```
{
  "case_id": "JC-2026-0042",
  "fraud_probability": 0.87,
  "threat_level": "systemic",
  "action": "file_case",
  "target": "did:DioChain:branch-shanghai-07",
  "evidence_summary": "...",
  "recommended_injunction": {
    "type": "freeze_withdrawal",
    "scope": "branch-shanghai-07",
    "duration_hours": 48
  }
}
```

4. **语义多数判决**：不要求 21 个节点的输出完全一致（AI 的输出本质上有随机性），而是要求在关键字段上达成 2/3+ 的"语义多数"：
   - `action` 字段：14+ 个节点输出 `file_case` 即触发立案。
   - `threat_level` 字段：14+ 个节点判定为 `systemic` 即确认系统级威胁。
   - 具体的 `fraud_probability` 数值可以有差异，但方向必须一致。

选择 21 个节点的原因：

- 足够多：单个恶意节点无法影响结果。

- 足够少：推理成本可控，响应时间在分钟级。
- 奇数：避免票数恰好持平。

9.2.4 紧急保全

立案后，如果情况紧急（资金正在被转移、攻击正在进行），AI 交警可以触发紧急保全措施：

- 调用涉事合约的**保全接口（Injunction Interface，见 9.4 节）**。
- 临时冻结特定提款功能或特定地址的资产。
- **冻结期有限**（默认 48 小时），到期后：
  - 如果法院已受理案件，冻结可延长。
  - 如果法院未受理，冻结自动解除，不可人为延长。

紧急保全并非判决——它是"先止血，再治疗"的应急措施。

9.2.5 成本与激励

司法系统如果免费，将被滥用。成本机制确保只有真正的威胁才会进入司法流程：

角色	胜诉	败诉/诬告
报警方	诉讼费全额退还 + 瓜分罚没保证金的一部分（Bounty）	诉讼费没收
AI 交警节点	获得推理费用补偿 + 少量激励	获得推理费用补偿（不惩罚）

Bounty 机制鼓励社区积极举报真正的系统性威胁。诉讼费没收机制阻止恶意诬告和刷单。

9.3 AI 法院（Adjudication）

AI 法院是司法原语的第二道防线——负责**深度审理和最终判决**。

9.3.1 庭审流程

AI 法院的审理流程采用对抗制设计：

1. 控方提交侦查报告

AI 交警（作为控方）提交完整的侦查报告：

- ZK 证据包。
- 资金流图谱（通过链上数据分析构建的资金流向可视化）。
- AI 交警的分析结论和推理过程。

2. 辩方提交反向证据

涉事方（或其委托的 AI Agent）有权提交辩护材料：

- 反向证据（同样通过 ZK-TLS 封装）。
- 对控方证据的质疑（指出证据链的缺漏或逻辑矛盾）。
- 替代解释（证明涉嫌行为有合理的商业原因）。

3. AI 法官深度推理

多个独立的 LLM 节点作为 AI 法官，对控辩双方的材料进行深度分析：

- AI 法官同样在 TEE 内运行。
- 每个 AI 法官节点使用可能不同的底层模型（模型多样性降低系统性偏见风险）。
- AI 法官不仅要得出结论，还要输出完整的**逻辑推导链（Chain of Reasoning）**——每一步推理都必须引用具体的证据。

9.3.2 判决机制

1. AI 法官出具"判决草书"

每个 AI 法官输出结构化的判决草书：

```
{
  "verdict": "guilty",
  "confidence": 0.92,
  "reasoning_chain": [
    {
      "step": 1,
      "claim": "涉事网点在 2026-02-15 至 2026-03-01 期间净额结算差异累计达 50,000
Token",
      "evidence_ref": "evidence_pkg_001, tx_hash_list_A"
    },
    {
      "step": 2,
      "claim": "差异模式与随机误差不一致，呈现单方向偏差（有利于网点方）",
      "evidence_ref": "statistical_analysis_report_B"
    }
  ],
  "recommended_penalty": {
    "slash_amount": 100000,
    "victim_compensation": 50000,
    "burn_amount": 30000,
    "bounty_amount": 20000
  }
}
```

2. 人类陪审团多签确认

AI 法官的判决草书并非最终判决。最终裁决由**高信誉人类陪审团（DAO 治理节点）** 多签确认。

引入人类审核的原因：

- **防止 AI 幻觉**：LLM 可能在推理过程中"编造"不存在的证据链接或逻辑跳跃。人类陪审团审核推理链的每一步是否有据可查。
- **防止对抗性攻击**：攻击者可能精心构造证据，使其在 AI 看来高度可信，但人类审核后发现逻辑破绽。
- **法律责任的锚定**：最终判决由人类签名确认，明确了法律责任主体。

陪审团的选拔：

- 从高信誉的 DAO 治理节点中随机抽选。

- 陪审成员需质押 Token，防止串通或怠工。
- 多签阈值（如 7/11）确认后，判决生效。

9.3.3 执行

判决生效后，执行是自动的：

1. 罚没（Slash）

- 从涉事网点的保证金中扣除罚没金额。
- 如果保证金不足以覆盖罚没，网点被强制关闭，剩余资产全部进入赔偿池。

2. 赔付

- 受害者的赔偿从罚没金额中拨付。
- 赔付通过托管支付合约自动执行，受害者无需额外操作。

3. 销毁

- 罚没金额中的一部分 Token 被永久销毁。
- 销毁比例由治理参数决定（可通过 DAO 投票调整）。
- 销毁是对系统性作恶的"公共惩罚"——作恶者不仅赔偿受害者，还要为损害整个网络的信任付出代价。

9.4 保全接口（Injunction Interface）

9.4.1 链级标准

保全接口是 DioChain 的一项**强制性链级标准**：所有核心协议（支付合约、ACTUS 合约、托管合约等）必须实现该接口。

这并非可选项。一个不可冻结的合约，就像一扇永远锁不上的门——在日常使用中没有区别，但在紧急情况下构成重大安全隐患。

9.4.2 接口定义

```
trait InjunctionInterface {
    /// 冻结特定地址的特定功能
    /// - address: 被冻结的 DID/地址
    /// - scope: 冻结范围（如 "withdrawal_only", "all_transactions"）
    /// - duration: 冻结时限（秒），到期自动解冻
    /// - proof: 授权证明（AI 交警共识或法院判决的签名）
    fn freeze(address: DID, scope: FreezeScope, duration: u64, proof:
JudicialProof);

    /// 解冻
    /// - address: 被解冻的 DID/地址
    /// - court_order: 法院判决书（证明解冻已被授权）
    fn unfreeze(address: DID, court_order: CourtOrder);
}
```

9.4.3 调用权限

谁可以调用 `freeze`：

- AI 交警共识：2/3+ 的 AI 交警节点达成语义多数后生成的 `JudicialProof`。
- AI 法院判决：陪审团多签确认后生成的 `JudicialProof`。

谁不可以调用 `freeze`：

- 任何单一实体——包括 DioChain 基金会、核心开发团队、L0 运营者。
- 没有经过 AI 交警共识或法院判决的任何请求。

这是一条红线。**保全接口的调用权不可被任何中心化权力垄断**。 这与传统金融中银行可以单方面冻结账户的模式有本质区别。

9.4.4 设计约束

1. 冻结必须有时限

- 每次 `freeze` 调用必须指定 `duration`。

- 到期后，冻结自动解除，无需任何额外操作。
- 如需延长冻结，必须重新经过司法流程（AI 交警再次共识或法院延期裁定）。

2. 冻结范围必须最小化

- `scope` 参数精确定义冻结的操作范围。
- 例如：只冻结提款功能，不冻结接收功能。只冻结特定合约的交互，不冻结全部资产。
- 禁止"全账户冻结"（除非法院判决明确要求）。

3. 所有冻结操作链上完全透明

- 每次 `freeze` 和 `unfreeze` 调用都记录在链上，全网可查。
- 记录内容包括：被冻结地址、冻结范围、冻结时限、授权证明的哈希。
- 任何人都可以审计冻结操作的合法性。

9.4.5 对抗审查的思考

保全接口的存在意味着 DioChain 并非一个"绝对不可审查"的系统。这是一个务实的权衡：

- 绝对不可审查的系统无法有效遏制违法行为。
- 绝对可审查的系统则可能沦为中心化权力的工具。
- DioChain 的选择：**受约束的、透明的、有限的可审查性**——只有经过去中心化共识的司法流程才能触发，操作完全链上可审计，冻结有时限且范围最小化。

第 10 章：合规插槽

DioChain 的目标是成为全球范围内的 Agent 金融基础设施。现实是：不同司法管辖区的金融监管要求差异巨大——有的要求严格的 KYC，有的禁止加密衍生品，有的有外汇管制，有的要求交易数据本地化。一个"要么全合规、要么全不合规"的系统注定只能服务于极少数市场。DioChain 的方案是在协议层预留**合规的可编程性**。

边界在哪里？ 合规插槽不是合规本身。DioChain 向监管方展示的是系统原生的合规可编程能力。插槽是预留的接口，具体的合规逻辑由部署方和网点运营方根据当地法规激活和配置。DioChain 协议层不做合规判断。

10.1 设计哲学

10.1.1 合规的可编程性，而非合规本身

传统金融系统的合规是硬编码的——每个银行都有一套与所在管辖区监管深度绑定的合规系统，跨国展业意味着重新构建整套合规栈。去中心化世界的另一个极端是完全无视合规——这条路走不远，任何达到一定规模的金融基础设施都无法回避监管。

DioChain 走的是第三条路：

核心思想：DioChain 协议层提供标准化的合规接口（插槽），但不填充具体的合规逻辑。合规逻辑由网点运营方根据当地法规"插入"这些接口。

10.1.2 面向监管者的价值主张

当 DioChain 的网点运营方与当地监管者对话时，其展示的核心信息是：

- “我们的系统在协议层预留了合规接口，各类合规要求都可以通过配置这些接口来实现。”
- “KYC 数据的处理方式、金融产品的限制范围、税务报告的生成格式——这些都是可配置的。”
- “所有合规配置都是链上可审计的——监管方可以实时验证网点确实启用了所要求的合规模块。”

这比"去中心化系统不需要合规"要务实得多，也比"为每个司法管辖区定制一套系统"要高效得多。

10.2 插槽类型

10.2.1 KYC/AML 插槽

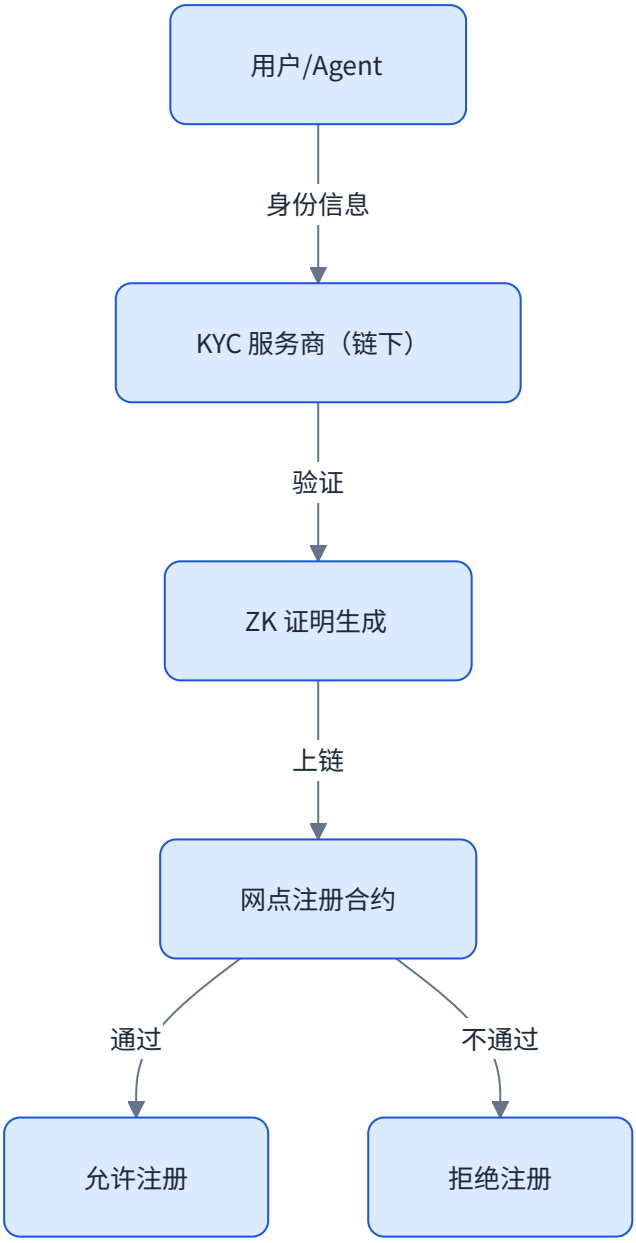
功能：控制谁可以在网点内注册和交易。

配置项：

配置	说明	默认值
`kyc_enabled`	是否启用 KYC	`false`
`kyc_provider`	KYC 服务提供商的合约地址	-
`kyc_level`	要求的 KYC 等级（基础/增强/完整）	-
`aml_screening`	是否启用 AML 筛查	`false`
`aml_provider`	AML 筛查服务的合约地址	-

工作方式：

1. 网点运营方根据当地法规启用 KYC 插槽。
2. 启用后，该网点内的 Agent/用户在注册时，必须通过指定 KYC 服务提供商的验证流程。
3. KYC 数据**不上链**——链上只存储验证状态的零知识证明。



隐私设计：

- 链上不存储姓名、身份证号、地址等个人信息。
- 链上只存储"是否通过验证"的 ZK 证明。
- 不同网点可以要求不同 KYC 等级——低风险业务只需基础 KYC，高风险业务需要完整 KYC。
- 用户在一个网点完成的 KYC 验证，可以在其他要求同等或更低等级的网点复用（前提是使用相同的 KYC 服务商，或服务商之间有互认协议）。

10.2.2 金融产品限制插槽

功能：控制特定网点内允许使用的 ACTUS 合约类型。

配置项：

配置	说明	默认值
`allowed_contract_types`	允许的 ACTUS 合约类型白名单	全部已启用类型
`blocked_contract_types`	禁止的 ACTUS 合约类型黑名单	空
`max_leverage`	最大杠杆倍数限制	无限制
`accredited_only_types`	仅限合格投资者使用的合约类型	空

应用示例：

- **场景 A**：某司法管辖区禁止加密衍生品交易。该管辖区内的网点将 `blocked_contract_types` 设置为 `[OPTNS, SWPP, FUTUR]`，禁止在该网点内创建期权、互换和期货合约。
- **场景 B**：某司法管辖区允许衍生品但要求合格投资者准入。该管辖区内的网点将 `accredited_only_types` 设置为 `[OPTNS, SWPP]`，只有通过增强 KYC（证明合格投资者身份）的 DID 才能创建这些合约。
- **场景 C**：某司法管辖区对杠杆有上限要求。该管辖区内的网点将 `max_leverage` 设置为 `5`，任何合约的有效杠杆不得超过 5 倍。

与 ACTUS 分阶段启用的关系：

- 第 7 章中描述的 ACTUS 分阶段启用策略是**全网级别**的——某些合约类型在全网范围内尚未启用。
- 金融产品限制插槽是**网点级别**的——即使某种合约类型在全网已启用，特定网点仍可因合规要求将其禁用。
- 两者是层级关系：全网未启用的，网点无法启用；全网已启用的，网点可以选择禁用。

10.2.3 税务报告插槽

功能：为网点提供交易数据的标准化导出能力，以满足当地税务申报要求。

配置项：

配置	说明	默认值
`tax_reporting_enabled`	是否启用税务报告	`false`
`report_format`	报告格式（OECD CRS / 自定义）	-
`report_frequency`	报告频率（实时/每日/每月/每季/每年）	-
`data_scope`	报告覆盖的数据范围	-

工作方式：

- 启用后，网点自动按照配置的格式和频率生成合规报告。
- 报告包含：交易列表、金额汇总、盈亏计算（基于 ACTUS 合约的精确现金流）。
- 报告导出为标准化格式，可直接对接当地税务系统。
- ACTUS 标准在这里发挥关键作用：因为所有金融合约的现金流都是结构化的，盈亏计算可以做到精确无误，无需人工估算。

10.2.4 跨境资金流动插槽

功能：控制跨网点资金转移的额度和审批流程。

配置项：

配置	说明	默认值
`cross_border_enabled`	是否允许跨网点转账	`true`
`single_tx_limit`	单笔跨境交易额度上限	无限制
`daily_limit`	每日跨境交易累计上限	无限制
`approval_required_above`	超过此金额需人工审批	无限制
`blocked_destinations`	禁止转入的网点列表	空

应用示例：

- **场景 A**：某司法管辖区有外汇管制。该管辖区内的网点设置 `daily_limit` 为 50,000 Token，超过部分需等待下一日。
- **场景 B**：某司法管辖区要求大额跨境汇款需人工审批。该管辖区内的网点设置 `approval_required_above` 为 10,000 Token，超过此金额的跨网点转账需要网点运营方手动审批。
- **场景 C**：某司法管辖区对特定地区实施限制。该管辖区内的网点将受限地区的网点加入 `blocked_destinations`。

注意事项：

- 跨境限制只能限制**从该网点发出**的跨境转账。网点无法控制其他网点向自己发起的转入。
- 如果 Agent 试图通过中间网点绕过限制，这涉及更复杂的合规执行——可以通过 L0 层的全局策略或多网点间的合规互认协议来解决，但这超出了单个插槽的范围。

10.3 插槽治理

10.3.1 谁有权激活/停用插槽

合规插槽的激活和配置涉及重大的权限问题。DioChain 定义了两级治理：

网点级治理（Branch-Level Governance）：

- 网点运营方有权为自己的网点激活、停用和配置合规插槽。
- 这是网点运营方的本地决策——他们了解当地法规，由他们负责合规。
- 网点运营方的合规配置变更记录在链上，全网可审计。

全网级治理（L0 DAO Governance）：

- L0 DAO 可以通过治理投票，强制全网启用某些**基础合规要求**。
- 例如：如果 DAO 投票决定"所有网点必须启用基础 AML 筛查"，则这一要求对全网所有网点生效，个别网点不可绕过。
- 全网级合规要求的设定需要高阈值投票（如 2/3 超级多数），防止轻率决策。

红线：不可被单一实体任意修改。

- DioChain 基金会不可以单方面要求某个网点启用或停用插槽。
- L0 运营者不可以在没有治理投票的情况下强制全网合规变更。
- 合规插槽的激活/停用必须通过链上治理流程，操作记录永久存档。

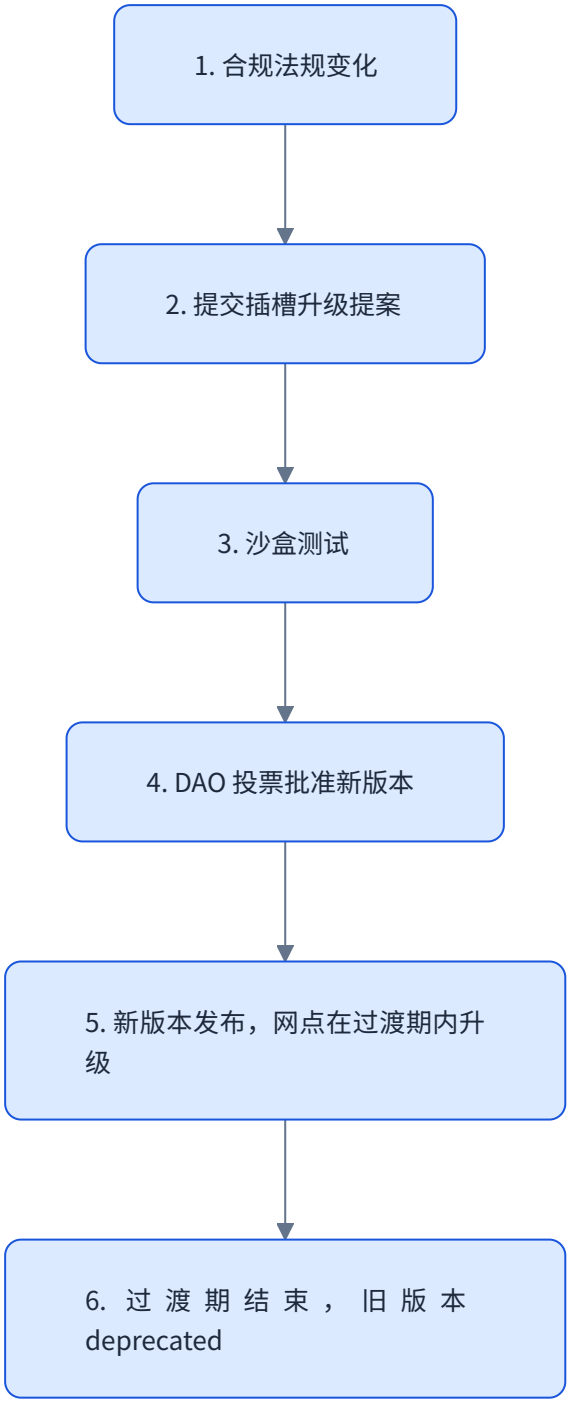
10.3.2 版本管理与升级机制

合规法规会变化，合规插槽也需要随之演进：

版本化插槽：

- 每个插槽类型都有版本号（如 `KYC-Slot v1.0`，`KYC-Slot v2.0`）。
- 新版本发布后，旧版本不会被强制替换——网点可以在过渡期内继续使用旧版本。
- 过渡期由 DAO 治理设定（如新版本发布后 6 个月内完成迁移）。

升级流程：



向后兼容性：

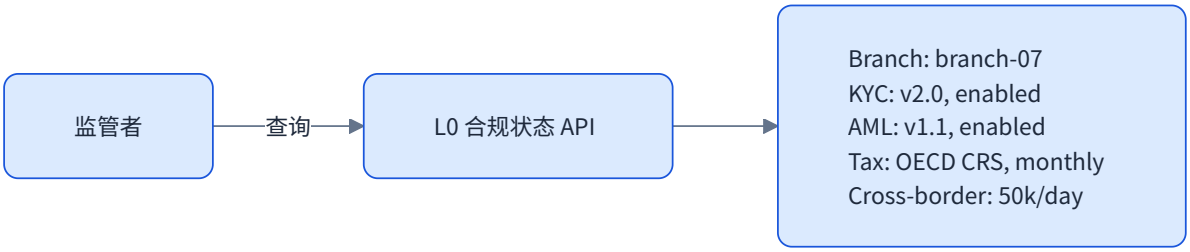
- 插槽升级应尽可能保持向后兼容——新版本应该是旧版本的超集。

- 如果不可避免地需要破坏性变更（Breaking Change），必须在提案中明确说明，且过渡期应相应延长。

10.3.3 插槽的可审计性

合规的可信度依赖于可审计性：

- 每个网点当前激活的插槽及其配置，链上可查。
- 每次插槽配置变更（激活、停用、参数修改）都记录在链上，包含：
  - 变更内容。
  - 变更时间。
  - 变更发起者的 DID。
  - 治理投票结果（如果是全网级变更）。
- 监管者可以通过标准化的查询接口，实时查看任意网点的合规状态——无需依赖网点的自我申报。



这种透明度是 DioChain 与监管者建立信任的基础。合规不再是一份纸质报告里的承诺，而是链上可实时验证的事实。

第 11 章：平行宇宙与跨链拓扑

DioChain 的原语集——经济、合约、身份、司法、合规——构成了一个完整的数字社会操作系统。但一个关键问题始终悬而未决：这套操作系统只能有一个实例吗？答案是否定的。DioChain 的设计从一开始就预设了多实例并行运行的可能性。本章阐述这一架构的逻辑基础、具体形态和跨链协作机制。

11.1 DioChain 不只是"一条链"

11.1.1 社会操作系统协议

DioChain 的本质不是"一条带有合规插槽的区块链"。它是一套**社会操作系统协议（Social Operating System Protocol）**——定义了 Agent 经济体运转所需的完整原语集和交互规范。

这个区分至关重要。一条链是一个实例；一套协议可以有无数个实例。Linux 内核是一套协议，Ubuntu、Red Hat、Android 是不同的实例——它们共享核心代码，但在用户界面、权限模型、应用生态上各不相同。DioChain 遵循同样的逻辑：核心代码开源，任何实体——主权国家、国际组织、企业联盟、DAO——都可以 fork 这套代码，在自己的治理框架下运行一个独立的 DioChain 实例。

11.1.2 多链平行宇宙

我们将这种架构称为**“多链平行宇宙”（Multi-Chain Parallel Universes）**。每个实例运行相同的原语集（经济原语、合约原语、身份原语、司法原语、合规插槽），但在治理参数、合规配置、身份绑定策略上可以完全不同。

这不是技术上的妥协，而是对现实世界的深刻尊重。

11.1.3 一个必须直面的现实

区块链行业长期以来有一种隐含的假设：去中心化必然优于中心化，主权国家是需要被绕过的障碍。这种假设在意识形态上有其吸引力，但在工程实践中是危险的。

现实是：**主权政府拥有巨大的资源禀赋和执行能力**。一个国家可以调动税收体系、法律系统、执法机构、教育网络、基础设施投资——这些能力在许多场景下远超任何去中心化乌托邦所能提供的。强制忽略这一现实，要么导致系统在合规压力下崩溃，要么导致系统只能在极小的灰色地带中苟延残喘。

DioChain 的选择是：**不对抗主权现实，而是将其纳入架构设计**。让去中心化的自由世界和主权管辖的秩序世界同时存在，各自发挥优势，并通过标准化的接口互联互通。

11.2 乌托邦链（Utopia Chain）

11.2.1 定位

乌托邦链是 DioChain 协议的**官方参考实例（Official Reference Instance）**，运行在"数字公海"上——不隶属于任何单一主权管辖区。它是完全去中心化的 Agent 自治经济体。

11.2.2 治理

乌托邦链采用 **PoS Token 加权投票** 作为治理机制：

- 投票权重与 $DIO 持有量成正比。
- 治理范围覆盖：协议升级、参数调整、全网合规基线、司法原语的规则库更新。
- 提案需要超过总质押量 2/3 的支持才能通过。

这是一种资本加权的治理模型——它的优势在于效率和利益绑定（持有者有动机维护系统健康），劣势在于财阀倾向（大户拥有更大话语权）。乌托邦链接受这一权衡，因为在没有主权背书的公海环境中，经济利益绑定是最务实的信任锚点。立法的完整流程——从链下游说到链上公投——由 diolaw 基础设施实现，详见第 12 章。

11.2.3 司法

乌托邦链完整运行第 9 章定义的 AI 司法系统：

- **AI 交警**：TEE 内运行的 LLM 节点网络，负责接警、取证、立案和紧急保全。
- **AI 法院**：多模型对抗制审理，输出结构化判决草书。
- **人类陪审团**：高信誉 DAO 治理节点多签确认最终判决。

没有传统意义上的最高法院或上诉机构——AI 法院的判决经人类陪审团确认后即为终局。

11.2.4 合规

乌托邦链的合规插槽处于**最小激活状态**：

- KYC 插槽：默认关闭。网点运营方可自行选择是否启用。
- AML 筛查：依赖市场涌现的风控服务，而非强制性合规要求。
- 金融产品限制：无。所有已启用的 ACTUS 合约类型均可自由使用。
- 跨境资金限制：无。

市场自我调节取代行政监管——风险由参与者自行评估和承担。

11.2.5 身份

乌托邦链采用**自主权 DID（Self-Sovereign DID）**：

- DID 的创建不需要任何现实世界身份的绑定。
- Agent 和人类用户可以匿名注册，仅凭密钥对即可获得完整的链上身份。
- 信誉系统完全基于链上行为记录，与现实身份无关。

11.3 主权链（Sovereign Chain）

11.3.1 定位

主权链是主权国家或区域性法律实体基于 DioChain 开源代码 fork 并运营的**本地化实例**。它继承了 DioChain 的全部原语集，但在治理、司法、合规和身份层面进行了深度定制，以符合当地法律和行政体系的要求。

11.3.2 身份

主权链的核心差异从身份层开始。DID 与国家公民身份系统**强绑定（Mandatory Binding）**：

- 每个 DID 必须关联一个经过政府认证的公民身份标识（身份证号、社会安全号等）。
- Agent DID 必须关联其所有者（自然人或法人）的已认证 DID。
- KYC 插槽强制启用，且 KYC 服务商必须为政府认可的机构。

11.3.3 治理

主权链可以采用 **Proof of Personhood（人格证明）** 治理模型，实现**一人一票**：

- 每个经过身份认证的公民 DID 拥有一票——不因财富差异而区别对待。
- 投票权与 Token 持有量脱钩。
- 这一模型依赖于强身份绑定——只有确保"一个人只有一个 DID"，才能防止女巫攻击（Sybil Attack）。

11.3.4 司法

主权链在司法层面拥有最大的灵活性：

- 可以完整保留 AI 交警和 AI 法院系统。
- 也可以**拔除 AI 法官模块，插入人类最高法院模块**——AI 法院的判决不再是终局，而是可以被上诉至由人类法官组成的最高仲裁庭。

- 司法原语的保全接口（Injunction Interface）保持不变，但调用权限可以扩展至经授权的政府执法机构。

11.3.5 合规

主权链**全面激活合规插槽**，按照当地法规进行配置：

- KYC/AML 插槽：强制启用，等级和服务商由监管机构指定。
- 金融产品限制：按当地法律设定白名单/黑名单。
- 税务报告：强制启用，格式和频率符合当地税务要求。
- 跨境资金流动：按当地外汇管制规定设定额度和审批流程。

11.3.6 经济

主权链可以发行**主权 Token 变体**：

- 锚定当地法币（如数字人民币、数字欧元）。
- Token 的铸造和销毁由中央银行或其授权机构管理。
- 与乌托邦链的 $DIO 之间通过跨链桥以市场汇率兑换。

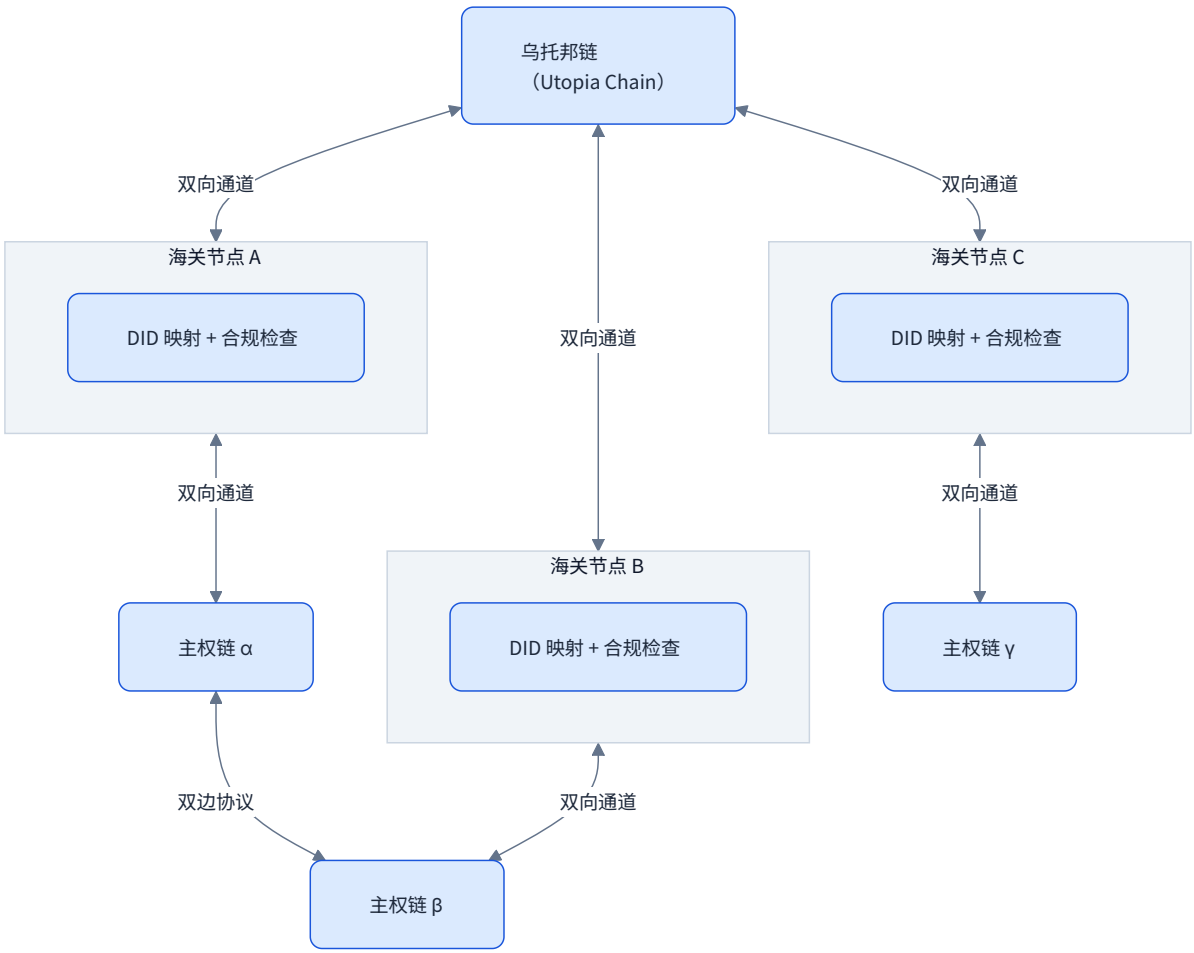
11.3.7 对比总览

维度	乌托邦链（Utopia Chain）	主权链（Sovereign Chain）
身份	自主权 DID，无需现实身份绑定	DID 强绑定国家公民 ID
治理	PoS Token 加权投票	Proof of Personhood，一人一票
司法	AI 交警 + AI 法院（终局）	可插入人类最高法院模块
合规	最小激活，市场自律	全面激活，按当地法规配置
经济	$DIO，可配置锚定	主权 Token 变体，锚定当地法币

11.4 跨链拓扑与 Agent 通行

11.4.1 拓扑结构

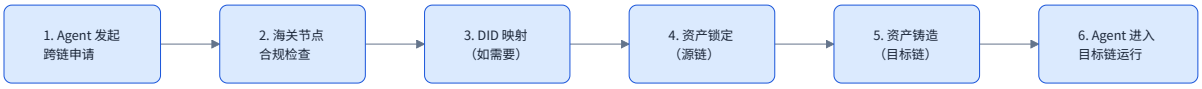
乌托邦链与各主权链之间不是层级关系，而是**网状拓扑**。每条链都是独立的对等节点，通过标准化的跨链协议互联。



跨链通信的核心枢纽是**海关节点（Customs Node）**——部署在两条链之间的中间层，负责身份映射、合规检查和资产锁定/释放。

11.4.2 Agent 签证与通关机制

Agent 从一条链迁移到另一条链，类比于自然人的跨国旅行。这一过程通过**签证/海关机制**（Visa/Customs Mechanism）实现：



详细流程：

1. **跨链申请**：Agent 向源链的海关节点提交跨链申请，声明目标链和迁移目的。
2. **合规检查**：海关节点验证 Agent 是否满足目标链的准入要求。例如，从乌托邦链迁移到某主权链，Agent 需要提供符合该主权链 KYC 要求的身份凭证。
3. **DID 映射**：如果目标链要求不同格式的 DID，海关节点执行 DID 映射（见 11.4.3 节）。
4. **资产锁定**：Agent 在源链上的资产被锁定在跨链桥合约中。
5. **资产铸造**：目标链上铸造等值的本地 Token（按市场汇率或固定汇率，取决于两链的协议）。
6. **入链运行**：Agent 获得目标链的 DID 和资产，开始在目标链上运行。

11.4.3 DID 映射

乌托邦链的自主权 DID 与主权链的强绑定 DID 之间可以建立**可选的双向映射**：

- **正向映射（乌托邦 → 主权）**：Agent 自愿将其乌托邦 DID 与主权链上的已认证 DID 关联。这是进入主权链的前提条件。
- **反向映射（主权 → 乌托邦）**：主权链用户可以选择将其已认证 DID 关联到一个乌托邦 DID，从而在乌托邦链上保留可验证的信誉背书。
- **映射是加密的**：映射关系通过零知识证明保护——第三方只能验证"某个乌托邦 DID 确实关联了某条主权链上的合法身份"，但无法获知具体的身份信息。

11.4.4 资产桥接

跨链资产转移通过**锁定-铸造-销毁-释放**（Lock-Mint-Burn-Release）模式实现：

- 从乌托邦链转至主权链：$DIO 在乌托邦链上锁定，主权链上铸造等值的本地 Token。
- 从主权链转回乌托邦链：主权 Token 在主权链上销毁，乌托邦链上释放等值的 $DIO。
- 每次跨链转移都经过海关节点的合规检查——主权链可以拒绝不符合其合规要求的入境资产。

11.5 链间关系：无绝对层级

11.5.1 对等而非从属

乌托邦链和主权链之间**没有层级关系**。乌托邦链不是"主链"，主权链不是"侧链"或"子链"。它们是同一协议的不同实例，在各自的治理框架下独立运行，地位完全对等。

这一设计拒绝了两种常见的架构陷阱：

- **Hub-Spoke 模型**：一条主链连接多条从属链。问题在于主链成为单点故障和权力中心。
- **Layer-1 / Layer-2 模型**：子链依赖母链进行最终结算。问题在于子链的主权性受到结构性限制。

DioChain 的平行宇宙是真正的平行——每条链拥有自己的共识、自己的治理、自己的司法、自己的经济。

11.5.2 各自的优势

乌托邦链的优势：

- **自由度**：无监管约束，创新可以在没有合规摩擦的环境中快速迭代。
- **全球触达**：不受任何单一司法管辖区的地理限制，天然面向全球。
- **抗审查**：没有单一实体可以关停系统或冻结账户（除非通过链上司法流程）。

主权链的优势：

- **法律确定性**：合约在当地法律框架下具有可执行性，纠纷可以诉诸传统法院。
- **政府资源**：可以接入国家级的基础设施——银行系统、支付网络、身份认证体系。

- **行政效率**：在需要集中决策的场景（如危机应对、系统升级）中，主权治理的响应速度远超去中心化投票。

11.5.3 完整光谱

两类链的共存形成了一个从**纯粹去中心化**到**完全主权管控**的完整光谱：

- 光谱的一端是乌托邦链：最大自由、最小管控。
- 光谱的另一端是高管控主权链：最大秩序、最小自由。
- 中间存在无数可能的配置：部分激活合规插槽的半自治链、多国联合治理的区域链、行业联盟运营的垂直链。

没有一种配置是"正确的"。不同的场景、不同的参与者、不同的风险偏好需要不同的配置。DioChain 协议的价值在于：它为所有这些配置提供了统一的技术基础。

11.6 监管现实与共存策略

11.6.1 乌托邦链的监管风险

乌托邦链运行在数字公海上，这意味着它可能面临来自特定国家的**制裁或封锁**：

- 部分国家可能禁止本国公民访问乌托邦链。
- 部分国家可能将乌托邦链上的资产视为非法持有。
- 传统金融机构可能拒绝与乌托邦链上的资产进行法币兑换。

这些风险是真实存在的，忽视它们是不负责任的。

11.6.2 主权链作为合法入口

主权链为 DioChain 生态提供了**合法的落地入口（Legal Entry Point）**：

- 在监管严格的司法管辖区，用户和 Agent 可以通过本地主权链参与 DioChain 生态——在完全合法合规的框架下使用相同的原语集。

- 主权链的存在使得 DioChain 技术可以被主权国家"招安"而非"封杀"——政府可以在自己的监管框架下运行 DioChain，而非试图消灭它。
- 对于监管者而言，一个可控的、可审计的、在本国法律框架下运行的 DioChain 实例，远优于一个无法控制的离岸系统。

11.6.3 互为备份

乌托邦链和主权链之间构成**互为备份**的关系：

- 如果某条主权链因政策变化被关停，该链上的用户可以通过跨链机制将资产和身份迁移至乌托邦链或其他主权链，避免资产归零。
- 如果乌托邦链在某些地区被封锁，用户可以通过当地主权链继续使用 DioChain 的核心功能。
- 在法规允许的条件下，主权链用户可以"出海"——将部分资产和活动迁移到乌托邦链上，享受更高的自由度和更广的全球触达。

11.6.4 务实而非软弱

这种共存策略可能被批评为"向权力妥协"。我们不同意这一评价。

去中心化的价值在于为个体提供选择权——而非强迫所有人接受同一种意识形态。一个只有乌托邦链的世界，实际上是在要求所有参与者接受"无监管"这一单一治理模式；一个同时提供乌托邦链和主权链的世界，才是真正把选择权交给了参与者。

地缘政治现实不是需要被否认的缺陷，而是需要被纳入设计的约束条件。 最优秀的工程不是无视约束的空想，而是在约束条件下寻找最大可行解。DioChain 的平行宇宙架构正是在主权现实这一不可消除的约束条件下，为 Agent 经济体寻找的最大可行解。

> **小结**：DioChain 不是一条链，而是一套社会操作系统协议。它支持两类实例并行运行——运行在数字公海上的乌托邦链（去中心化自治、Token 加权治理、AI 司法、最小合规）和主权国家 fork 运营的主权链（强身份绑定、一人一票治理、可插拔人类司法、全面合规）。两者之间没有层级关系，通过海关节点实现 Agent 通行、DID 映射和资产桥接。这一架构不是对去中心化理想的妥协，而是对地缘政治现实的工程化回应——让自由与秩序共存，让参与者自己选择所处的位置。

第 12 章：官方基础设施矩阵与闭环

> 一个社会如果只有法律却没有警察，只有银行却没有身份证，只有议会却没有证据链——它就不是一个真正可运转的社会。DioChain 的四件套基础设施，不是"开发者工具"，而是 Agent 社会的存在性前提。

12.1 从"工具"到"基础设施闭环"

前述章节分别阐述了 DioChain 的三层架构、原语集、算力市场和冷启动策略。读者可能将 aibank、agenter 理解为"官方提供的开发者工具"——一组 SDK 和 CLI，方便接入链上功能。这种理解是不完整的。

本章要论证的核心观点是：**aibank、agenter、diolaw、dioproof 四者构成一个不可拆分的基础设施闭环**。它们不是"工具"，而是 Agent 社会得以运转的结构性前提——正如人类社会不可能只有市场没有户籍、只有法院没有证据制度。

四个组件分别覆盖 Agent 社会运转的四个必要环节：

组件	核心职责	人类社会类比
aibank	身份管理 + 资产持有 + 网点接入 + 合约交互	户籍局 + 银行
agenter	需求理解 + 任务拆解 + 多模型调度 + 链上结算	全能秘书 + 经纪人
diolaw	提案发起 + 游说投票 + 链上公投 + 规则演进	议会 + 立法程序
dioproof	证据采集 + ZK 封装 + 链上存证 + 司法调用	公证处 + 司法鉴定

务实主义声明：DioChain 并不追求意识形态层面的"绝对去中心化"。在这四个组件的设计中，我们在效率和安全性有明确需要的地方采用了中心化组件——例如 agenter 的模型调度层、aibank 的密钥托管选项、diolaw 的链下投票聚合。务实主义的立场是：**哪里需要中心化就中心化，但中**

心化的边界必须被严格定义、链上可审计、可被去中心化方案替代。 这不是妥协，而是工程判断。

四者形成的闭环逻辑如下：

```
身份管理（aibank）→ 任务执行（agenter）→ 规则制定（diolaw）→ 争议举证（dioproof）→ 回到身份管理
```

移除任何一个环节，Agent 社会都无法完成"治理-执行-举证"的完整循环。没有 aibank，Agent 无法获得身份和资产；没有 agenter，普通用户无法将意图转化为链上行为；没有 diolaw，规则无法随社会演化而更新；没有 dioproof，链下证据无法进入司法流程。

12.2 aibank：标准化接入工具

aibank 是一个钱包 SDK/CLI（通过 `npx aibank` 启动），是人类和 AI Agent 接入 DioChain 的标准化工具。第 4 章已经介绍了它作为 L2 居民桥梁的基本功能。这里强调它在基础设施闭环中的位置。

12.2.1 功能覆盖

aibank 的功能超出传统钱包的范畴，但它的本质仍然是**工具**——DioChain 才是操作系统，aibank 是接入这个操作系统的标准化通道。

它覆盖 Agent 社会生活的五项基础需求：

1. **身份管理**：DID 的创建、密钥轮换、多签授权、自主等级声明。
2. **网点接入**：L1 网点发现、连接管理、故障切换。
3. **算力调度**：算力市场的查询、比价、调用和结算。
4. **合约交互**：ACTUS 合约的创建、签署、执行、查询。
5. **资产管理**：$DIO 的持有、转移、质押、查询。

这五项功能是串联的——没有身份就无法接入网点，没有网点就无法调用算力，没有算力就无法执行合约。aibank 的价值在于用一套统一的 API 覆盖了 Agent 从"出生"到"参与经济活动"的全链路，开发者不需要分别对接五套不同的接口。

12.3 agenter：面向普罗大众的超级调度器

agenter（通过 `npx agenter` 启动）的目标用户不是开发者，而是**普罗大众**。它的核心能力是将人类的模糊自然语言意图，转化为精确的链上经济行为。

agenter 不自研大模型，而是接入市场上最成熟的 Agent 工具——Claude Code、Codex、Gemini 等——通过 DioChain 算力市场按量付费调用。它将复杂需求拆解为分步骤的执行方案，调度多个成熟 Agent 工具协作完成，最终通过 aibank 在链上结算。用户全程无需了解 DioChain 的技术细节。

agenter 在基础设施闭环中的关键位置在于：**它是算力市场的第一个也是最大的买方**。 每一次用户与 agenter 的交互，都会触发对算力市场的真实消费——需求理解、方案生成、风险评估、结果呈现，每一步都消耗算力并以 $DIO 结算。这为算力市场创造了持续的、与投机无关的内生需求（详见第 14 章冷启动策略）。

12.4 diolaw：立法基础设施

AI 社会需要法律。这个论断在直觉上显而易见——人类社会的一切制度都建立在法律之上，AI Agent 社会没有理由例外。但在区块链领域，“立法"长期被简化为"DAO 投票”。DioChain 认为这远远不够。diolaw 是一套完整的立法基础设施，覆盖从提案发起到法律生效的全流程。

12.4.1 设计哲学

为什么 Agent 社会需要立法机制?

DioChain 的原语集（经济原语、合约原语、身份原语、司法原语、合规插槽）定义了 Agent 社会的初始规则。但社会是演化的——新的经济模式会出现，新的风险会涌现，旧的规则会过时。如果规则不能随社会演化而更新，系统将走向僵化或混乱。

传统区块链对此的回应是"DAO 投票"——发一个提案，持币者投票，多数通过即生效。这种机制有两个根本缺陷：

1. **链上投票成本高昂**：每一次投票都是一笔链上交易，需要 gas fee。如果每个提案都要求全网数十万 DID 逐一链上投票，成本和延迟都不可接受。
2. **投票质量低下**：绝大多数持币者不会认真阅读提案内容。他们要么跟风大户投票，要么干脆不投。这导致 DAO 投票的实质参与率通常低于 10%。

diolaw 的设计采用了一种更务实的**混合立法模型**：链下游说 + 链上公投。链下游说阶段解决"让提案被充分讨论和评估"的问题；链上公投阶段解决"最终决策的合法性和不可篡改性"的问题。

12.4.2 链下游说阶段

1. 提案提交

任何持有有效 DID 的实体（人类或 Agent）都可以通过 diolaw CLI 提交立法提案：

```
diolaw submit-proposal \
  --title "将算力供应商最低保证金从 100K 调整为 50K DIO" \
  --category "economic-parameter" \
  --full-text "./proposal-2026-0042.md" \
  --sponsor-did "did:dio:human:0x3a7f..."
```

提案内容必须包含：变更内容的精确描述、变更理由、预期影响分析、回滚方案。

2. AI 议员机制

diolaw 的核心创新在于 **AI 议员（AI Delegate）** 制度。

在 DioChain 的治理模型中，普通用户可以将自己的投票权委托给 AI 议员。AI 议员是运行在链上的自主 Agent，它们的职责是代表委托人阅读、评估和投票。一个 AI 议员可能代表数千个用户的投票权。

当一个新提案被提交后，提案发起人需要"游说"这些 AI 议员——让它们阅读提案内容，理解提案的利弊，并决定是否支持。

3. 关键经济机制：游说成本由提案发起方承担

这是 diolaw 最核心的设计：**AI 议员"阅读和思考"的算力成本，完全由提案发起方（或其受益方）承担**。

逻辑如下：

- AI 议员评估一个提案，需要调用 LLM 进行深度推理——理解提案内容、分析利弊、与历史提案对比、预测潜在影响。
- 这些推理消耗算力，算力消耗需要 $DIO 支付。
- 如果让 AI 议员自己承担评估成本，它们会倾向于忽略提案（理性的经济行为）。
- 因此，diolaw 规定：提案发起方必须为游说过程中的算力消耗买单。

这一设计的妙处在于：**它将立法活动与算力市场直接挂钩**。 推动立法是有利可图的——如果一个提案对某个利益群体有利，该群体就有动机出资购买算力（食物）来游说 AI 议员。这与人类社会的游说机制在经济逻辑上是同构的，只是媒介从"律师费和公关费"变成了"算力费"。

4. 链下投票聚合

游说过程中，AI 议员的投票在链下聚合。链下投票的优势是：

- 零 gas 成本——投票不上链，不消耗结算资源
- 高速聚合——投票结果实时可查
- 隐私保护——投票过程中，各议员的投票倾向可以选择性加密，防止"跟风投票"

链下投票的结果由一组可信聚合节点维护，聚合节点定期发布投票摘要的 Merkle Root，供任何人验证。

12.4.3 链上公投阶段

链下投票不是无限期的。diolaw 采用**竞争性榜单机制（Leaderboard Mechanism）**：所有活跃提案按已获得的支持票数排名，当一个提案的排名和票数同时满足预设的榜单门槛时，自动触发链上公投。这种设计让多个提案在链下阶段形成竞争——有限的 AI 议员注意力和算力资源会自然流向最有价值的提案。

链上公投的双轨制：

diolaw 是一个**协议级**的立法基础设施，运行在不同类型的链上时，采用不同的公投机制——由链类型决定，而非由提案类型决定：

维度	乌托邦链（Utopia Chain）	主权链（Sovereign Chain）
投票权重	PoS 加权（$DIO 持有量）	PoP 一人一票（DID 绑定公民身份）
身份基础	自主权 DID，无需现实身份	DID 与公民身份证强绑定
通过阈值	持币量加权 2/3 多数	参与者 2/3 多数
设计理念	利益相关者主导（资本加权）	公民权利保障（人头平等）

乌托邦链在数字公海上运行，没有主权身份背书，$DIO 持仓量是唯一可靠的利益绑定信号，因此采用 PoS 加权投票。主权链的 DID 与公民身份证强绑定，每个 DID 唯一对应一个自然人，具备防女巫攻击的基础，因此可以采用 PoP 一人一票——这与主权国家的选举逻辑一致。

链上公投流程：

12.4.4 AI 游说经济学

游说市场的定价机制遵循以下基本公式：

```
游说总成本 = 提案复杂度系数 * 需游说的 AI 议员数量 * 单次评估算力消耗
```

提案复杂度系数：一个修改单一参数的简单提案（如"将手续费率从 0.1% 调整为 0.08%“），AI 议员的评估成本较低；一个涉及多个原语交互的复杂提案（如"引入新的 ACTUS 合约类型”），评估成本显著上升。复杂度系数由 AI 议员在评估过程中自行决定——它们可以选择投入更多或更少的推理资源。

防垃圾机制：diolaw 设定了提案的最低算力投资门槛。提案发起方必须预存一定数量的 $DIO 作为"游说基金"（Lobbying Fund），低于此门槛的提案不会被系统接受。这防止了零成本的垃圾提案泛滥。

防贿赂机制：所有投票记录（无论链上还是链下）最终都会上链存档，全网可查。AI 议员的投票模式可以被公开审计——如果某个 AI 议员的投票与其委托人的利益持续背离，委托人可以撤回委托。这创造了一个自我纠偏的市场。

激励对齐：游说经济学的核心逻辑是——只有对足够多利益相关者有利的提案，才能募集到足够的游说资金。如果一个提案只对少数人有利但对多数人有害，它很难筹集到足够的算力来说服代表多数人利益的 AI 议员。这在经济层面实现了"多数决"的效果，且比传统投票更精细——不是简单的"赞成/反对"二元选择，而是通过资源投入量反映利益强度。

12.4.5 主权链适配：模块可插拔

diolaw 的架构自始至终按照**模块化立法框架**而非单体系统来设计。前述 12.4.1–12.4.4 描述的是 diolaw 在乌托邦链上的默认实现——AI 议员、算力游说、PoS 加权公投——这套实现适用于"数字公海"场景。然而，当 DioChain 以主权链形态部署在某一国家或地区的司法管辖区内时，该主权链的运营方（通常是政府机构或其授权实体）可以对 diolaw 的治理逻辑进行内核级模块替换。

具体而言，以下四个模块被设计为可插拔的：

1. **链下游说阶段**：乌托邦链上的开放游说市场可以被替换为政府授权的立法提案通道。在这种模式下，只有经过行政审批或议会常委会认可的提案才能进入立法流程，取代原有的"任何 DID 皆可提案"机制。

2. **AI 议员机制**：AI Delegate 可以被替换为人类民选代表或政府任命的立法委员。投票权委托逻辑从"用户委托给 AI Agent"变为"公民委托给民选议员"，与传统代议制无缝对接。
3. **链上公投机制**：乌托邦链的 PoS 加权投票在主权链上天然切换为 PoP 一人一票（因为主权链的 DID 与公民身份证强绑定，具备防女巫攻击的基础，详见 12.4.3 双轨制说明）。
4. **榜单门槛机制**：竞争性榜单排名可以被替换为议会委员会审议制度。提案是否进入公投不再由链下投票的自然排名决定，而由专门委员会根据法定程序审核决定。

可插拔边界的严格定义：模块可插拔不意味着协议层面的分裂。diolaw 的协议接口——包括提案数据格式（Proposal Schema）、投票数据结构（Vote Record）、链上记录格式（On-chain Legislation Entry）——在所有链类型上保持一致。变化的仅仅是每个阶段背后的治理逻辑实现。这意味着为乌托邦链上的 diolaw 构建的工具（提案浏览器、投票分析面板、立法历史查询接口）可以在主权链上以极低的适配成本运行。协议层不变，实现层可换——这是 DioChain 内核模块插拔哲学在治理领域的具体体现。

务实生存策略：这种设计并非技术上的多余。第 11 章 11.6.2 节已经论证了 DioChain 的监管共存策略——与其被主权国家视为不可控的威胁而遭到封禁，不如提供一种"可驯化"的接入方式，让政府能够在保持自身立法主权的前提下采纳 DioChain 的技术基础设施。diolaw 的模块可插拔设计正是这一策略在立法层面的工程落地：政府保留对"谁可以提案、谁可以投票、什么条件下公投"的完整控制权，而 DioChain 提供的是提案的标准化格式、投票的密码学可验证性、立法记录的不可篡改存储。技术为主权服务，而非取代主权——这是 DioChain 被政府"招安"而非"封杀"的关键前提。

12.5 dioproof：证据基础设施

第 9 章介绍了 AI 交警和 AI 法院如何处理系统级威胁。但司法系统的有效性依赖于一个前提：**可信的证据**。在一个链上链下交织的世界里，大量关键证据存在于链下——银行转账记录、聊天记录、网页内容、API 响应。dioproof 的职责是将这些链下证据转化为链上可验证的密码学证据包。

12.5.1 ZK-TLS 技术核心

dioproof 的技术基座是 **ZK-TLS（Zero-Knowledge TLS）**。其原理在第 9 章已有简要介绍，这里做更完整的阐述。

ZK-TLS 允许证据提供者在不暴露原始数据全文的前提下，证明以下事实的组合：

- **来源真实性**：该数据确实来自某个特定的 HTTPS 服务器（通过 TLS 证书链验证）
- **时间确定性**：该数据是在某个特定时间窗口内获取的（通过 TLS 会话时间戳）
- **内容断言**：该数据满足某些特定条件（通过零知识证明，如"银行余额大于 X"而不泄露具体余额）

12.5.2 支持的证据类型

dioproof 支持四类链下证据的 ZK 封装。每类证据经过采集后，均输出统一格式的 Evidence Package（.zkpkg），其 ZK 证明覆盖来源真实性、时间确定性和内容断言三个维度。

证据类型	典型来源	ZK 证明覆盖范围
聊天记录	即时通讯平台的会话数据	证明特定对话确实发生且包含特定内容片段，不泄露对话全文
银行转账凭证	银行网站的交易记录页面	证明特定银行在特定时间展示了特定交易记录，可选择性隐藏具体金额
网页快照	任意 HTTPS 网页	将网页在特定时刻的状态固定为密码学证据，防止被举报方事后篡改内容
API 响应	RESTful / GraphQL 等服务端点	证明特定 API 在特定时间返回了特定响应（或未在承诺时限内响应），适用于算力市场 SLA 争议

具体的采集方式（CLI、SDK 或浏览器插件）属于实现层面的技术选型，将在工程阶段确定。架构层面的关键约束是：无论采集入口如何变化，输出的 .zkpkg 格式和 ZK 验证逻辑保持统一。

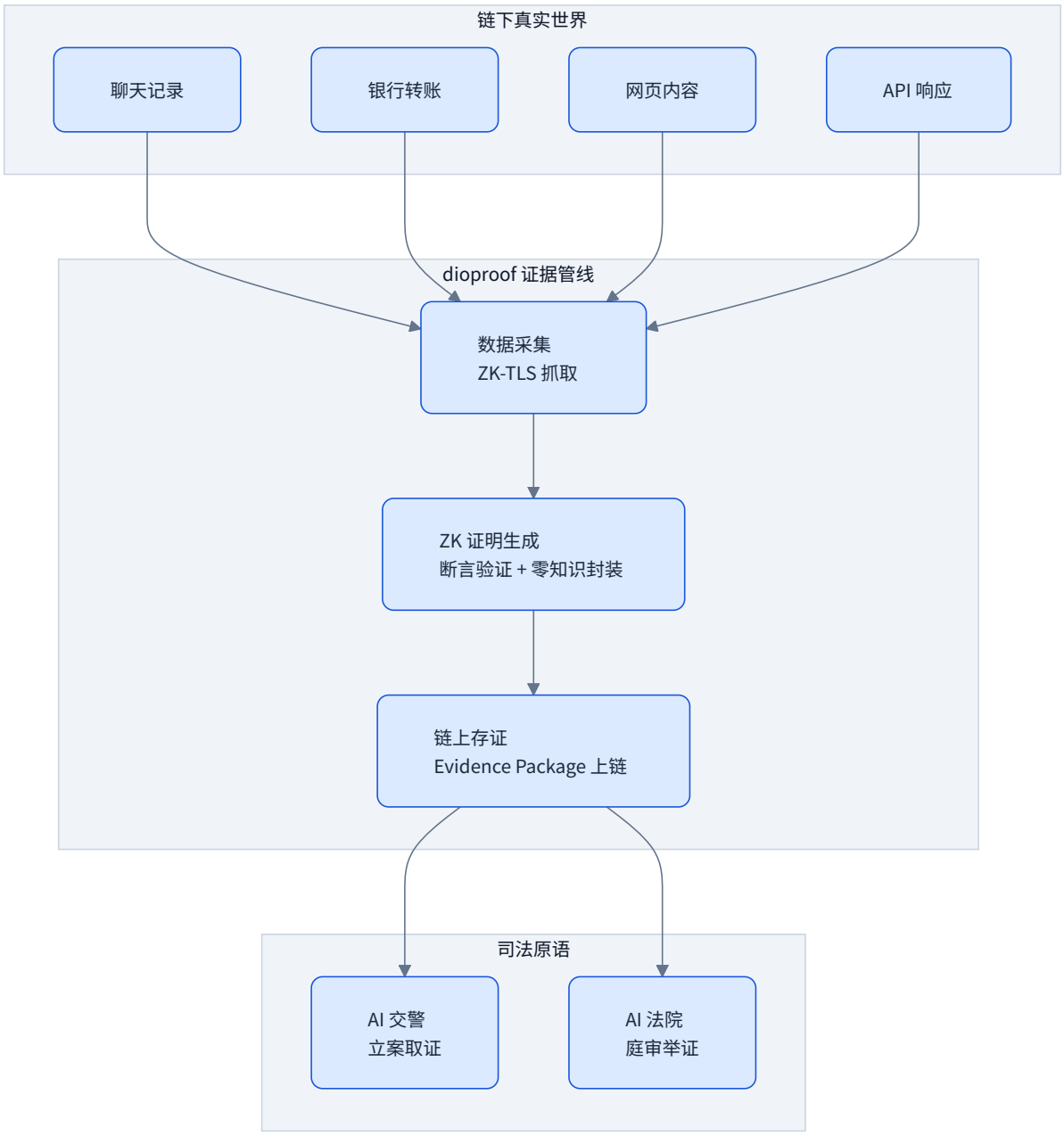
12.5.3 证据包格式

所有类型的证据都被封装为统一格式的 **Evidence Package（.zkpkg）**：

```
EvidencePackage {
  package_id: Hash,                    // 全局唯一标识
  evidence_type: EvidenceType,         // chat | bank-transfer | webpage |
api-response
  capture_timestamp: DateTime,         // 采集时间
  source_proof: {
    tls_cert_chain: [Certificate],     // TLS 证书链（证明来源）
    server_identity: String,           // 服务器域名
    session_timestamp: DateTime,       // TLS 会话时间
  },
  content_proof: {
    assertion: String,                 // 内容断言（自然语言描述）
    zk_proof: ZKProof,                // 零知识证明（证明断言为真）
    data_hash: Hash,                  // 原始数据哈希（不可逆）
  },
  collector_did: DID,                  // 采集者身份
  signature: Signature,                // 采集者签名
}
```

12.5.4 与司法原语的集成

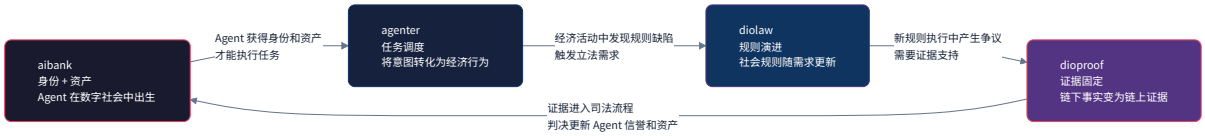
dioproof 采集的证据包直接对接第 9 章定义的司法原语：



在司法流程中，无论是控方还是辩方，都可以通过 dioproof 提交链下证据。AI 交警在侦查阶段使用 dioproof 固定证据现场；AI 法院在庭审阶段验证控辩双方提交的证据包的密码学有效性。

12.6 闭环：四件套如何形成自洽系统

四个基础设施组件不是并列关系，而是**首尾相接的闭环关系**。每一个组件的输出是下一个组件的输入，每一个组件的功能依赖于其他组件的存在。



12.6.1 闭环的必要性论证

移除任何一个组件，闭环立即断裂：

没有 aibank：Agent 无法获得身份，无法持有资产，无法接入网点。agenter 无法为用户执行任何链上操作。diolaw 的投票机制失去身份基础。dioproof 的证据包无法关联到具体实体。整个系统瘫痪。

没有 agenter：普通用户无法将意图转化为链上行为。DioChain 变成一个只有开发者才能使用的技术平台，丧失了面向大众的能力。算力市场失去最大的内生买方，冷启动无法完成。

没有 diolaw：DioChain 的规则固定在创世时刻，无法随社会演化而更新。当新的经济模式出现、新的风险涌现时，系统只能依赖核心开发团队的硬分叉来应对——这正是 DioChain 试图超越的中心化治理模式。

没有 dioproof：司法原语的 AI 交警和 AI 法院失去了链下证据来源。在一个链上链下交织的世界里，大量争议涉及链下行为——如果链下证据无法以密码学可验证的方式进入司法流程，司法原语的管辖范围将被严重压缩，系统性威胁无法被有效遏制。

12.6.2 与人类社会的同构映射

这个四组件闭环在结构上与人类社会的基础制度高度同构：

公民注册（户籍/身份证）→ 经济活动（工作/交易）→ 立法参与（选举/提案）→ 法律争议（诉讼/取证）→ 回到公民权利

- aibank 对应户籍制度和银行体系——没有身份就无法参与社会。
- agenter 对应经纪人和专业服务——普通人通过中介参与复杂的经济活动。
- diolaw 对应立法机构——规则由社会成员的集体意志决定。
- dioproof 对应公证制度和司法鉴定——争议需要可信的证据来裁决。

这种同构不是偶然的。Agent 社会的治理需求与人类社会在本质上是相同的——身份、经济、立法、司法是任何社会运转的四根支柱。DioChain 的贡献不在于发明了新的社会结构，而在于将这些经过千年验证的社会结构用密码学和 AI 技术重新实现，使其能够在数字世界中以代码的确定性运行。

12.6.3 多链并行与基础设施演进的反馈循环

上述闭环描述的是单个 DioChain 实例内部的结构。在第 11 章定义的多链并行框架下，不同治理模型的链实例可能根据各自的社会需求，对四件套基础设施进行差异化配置或扩展——例如，某条主权链可能为 dioproof 增加符合本地司法管辖要求的证据类型，或在 diolaw 中引入适配本国立法程序的投票规则。当这些局部创新被验证有效后，它们可以通过跨链治理协议回馈到其他链实例，形成"局部实验-验证-回移"的正向循环。基础设施矩阵因此不是一成不变的静态设计，而是在多链生态的竞争与协作中不断进化的动态系统。

小结

aibank、agenter、diolaw、dioproof 四个基础设施组件，将 DioChain 从"一个区块链协议"提升为"一个完整的社会操作系统"。

区块链协议提供的是底层能力——共识、结算、智能合约。社会操作系统提供的是完整的社会运转框架——身份管理、经济调度、规则演进、证据固定。前者是必要条件，后者是充分条件。DioChain 的原语集（经济、合约、身份、司法、合规）定义了社会的初始规则；四件套基础设施

确保这些规则能够被执行、能够被更新、能够在争议中得到公正裁决。

务实主义贯穿始终。我们不追求每一个组件都"去中心化到极致"，而是追求**每一个组件都能有效地完成其职责**。在需要中心化的地方（如 agenter 的模型调度、diolaw 的链下投票聚合），我们坦然采用中心化方案，但严格约束其边界并保留去中心化替代路径。在需要去中心化的地方（如 aibank 的 DID 管理、dioproof 的证据验证、diolaw 的链上公投），我们坚持密码学保障的不可篡改性。

这不是理想主义的妥协，而是工程实践的智慧。一个能运转的系统，永远比一个"理论上完美但无法启动"的系统更有价值。

第 13 章：去中心化算力交易中心

> 算力是 Agent 的食物。DioChain 的定位是构建算力交易市场，而非算力生产设施。

13.1 定位：交易协议，而非算力云

DioChain 的算力交易中心是一个**去中介化的算力聚合平台**，本质上更接近交易市场的角色——自身不生产商品，只提供交易协议。

需要明确的是，DioChain **不做**以下事情：

- 不代理 API 调用（DioChain 不充当中间商）
- 不托管模型（DioChain 不提供算力云服务）
- 不管理推理节点（DioChain 不承担编排调度职能）

DioChain **做**的事情：

- 定义供应商的注册与声明标准
- 提供链上可验证的 SLA 与结算协议
- 让 Agent 能够自主发现、比较、切换算力供应商
- 让算力供应商通过质押保证金获取信用背书

核心类比：

概念	传统类比	DioChain 实现
模型供应商	在交易平台开店的商家	来链上开设算力网点，缴纳保证金
用户/Agent	在交易平台购买服务的消费者	按量付费调用，自主比价
DioChain	交易平台	只提供交易协议和争议处理机制
中心化算力聚合平台	品牌代理商（赚差价）	被去中介化的对象

与传统中间商模式的关键区别：在中心化算力聚合平台模式中，用户的请求经过平台代理转发，平台赚取价差。在 DioChain 模式中，用户直接与供应商交互，链只负责结算和争议仲裁——**没有中间商赚差价**。

13.2 供应商接入协议

13.2.1 模型注册标准

任何算力供应商要在链上"开店"，必须提交结构化的**能力声明（Capability Declaration）**：

```
CapabilityDeclaration {
  provider_did: DID,                    // 供应商的去中心化身份
  model_id: String,                     // 模型标识（如 "llama-3.1-70b"）

  // 能力描述
  capabilities: {
    task_types: [TaskType],             // 支持的任务类型（chat, completion,
embedding, image, code...）
    context_window: u32,                // 上下文窗口大小（token 数）
    languages: [LanguageCode],          // 支持的语言
    modalities: [Modality],             // 支持的模态（text, image, audio,
video）
    max_output_tokens: u32,             // 最大输出 token 数
  },

  // 定价模型
  pricing: PricingModel,                // 见下文

  // SLA 声明
  sla: {
    latency_p50_ms: u32,               // P50 延迟承诺
    latency_p99_ms: u32,               // P99 延迟承诺
    availability: Decimal,              // 可用性承诺（如 0.999）
    throughput_tps: u32,                // 吞吐量承诺（tokens per second）
  },

  // 端点信息
  endpoint: URL,                        // API 端点
  api_schema: APISchema,                // 兼容的 API schema（通用兼容格式等）
}
```

13.2.2 定价模型

支持三种定价模式，供应商可自由选择：

模式	适用场景	示例
Per-Token	标准文本推理	输入 0.5 $DIO / 1M tokens，输出 1.5 $DIO / 1M tokens
Per-Request	固定任务（如 embedding）	1 $DIO / 1000 requests
Per-Second	长时间占用的任务（如 fine-tuning）	10 $DIO / GPU-hour

定价由供应商自行设定，市场竞争自然收敛价格。链不设价格管制。

13.2.3 算力网点与金融网点的关系

算力网点是 L1 网点的一种特化形态：

- **合一模式**：一个实体同时运营金融网点和算力网点。例如，一家 AI 公司既提供模型推理服务，又提供 $DIO 的兑换和清算。这种模式下，该网点可以内部结算算力费用，减少跨网点清算开销。
- **独立模式**：纯算力供应商只开设算力网点，不参与金融清算。算力费用通过 L0 结算层与其他金融网点完成跨网点清算。

两种模式在技术上没有优劣之分，取决于供应商的业务定位。

13.2.4 保证金要求

算力网点与金融网点一样，必须质押保证金：

- **最低保证金**：由 L0 设定基准线，与 SLA 承诺挂钩（承诺越激进，保证金越高）
- **罚没条件**：SLA 违约时自动罚没（见 13.3.3 争议处理）
- **保证金释放**：供应商主动退出后，保证金进入冷却期（如 7 天），期间处理未结算的争议

这确保供应商拥有切实的经济利益绑定（skin in the game）——虚假的 SLA 声明会导致保证金被罚没。

13.3 算力路由与结算

13.3.1 Agent 自动比价

当一个 Agent 需要调用 AI 模型时，它不需要绑定单一供应商。链上的能力声明注册表是公开可查询的，Agent 可以基于多维度自动选择最优供应商：

```
RouteDecision = f(price, latency, quality, availability, reputation)
```

路由策略由 Agent 自行决定（或由其开发者预配置），常见策略包括：

- **最低价格优先**：适合对延迟不敏感的批处理任务
- **最低延迟优先**：适合实时对话场景
- **加权综合评分**：考虑价格、延迟、历史 SLA 达标率的综合排序
- **冗余调用**：同时调用多个供应商，取最快返回的结果（以额外成本换取可靠性）

Agent 的路由决策完全在客户端（L2）完成，链不强制任何路由策略——这符合"定义原语不定义应用"的设计哲学。

13.3.2 流支付结算

算力消费的结算采用**流支付（Streaming Payment）**机制，按 token（词元）级别微支付：

1. Agent 在调用前，锁定一笔预付款到 Payment Channel
2. 供应商每生成一个 token，Agent 签署一笔微支付凭证
3. 调用结束后（或 Payment Channel 超时），供应商提交最终凭证结算

这种机制的优势：

- **消除信任风险**：供应商不用担心 Agent 用完不付钱，Agent 也不用担心供应商收钱不干活
- **按量精确计费**：不存在"买了 100 万 token 额度但只用了 3 万"的浪费

- **支持中断恢复**：如果调用中途断开，已生成的 token 已经结算完毕

13.3.3 争议处理

SLA 未达标时的争议处理流程：

1. **自动检测**：链上记录每次调用的请求时间戳和响应时间戳，自动计算实际延迟和可用性
2. **自动赔付**：如果供应商在统计窗口内的 SLA 达标率低于声明值，触发自动赔付（从保证金中扣除）
3. **阈值罚没**：如果 SLA 持续严重违约（如可用性低于 90%），触发保证金大额罚没 + 临时下架
4. **申诉机制**：供应商可以提交证据申诉（如网络层面的不可抗力），由 AI 司法系统裁决

争议处理的关键设计原则：**以链上可验证数据为准**。响应时间、可用性等指标都记录在链上，不存在模糊的灰色地带。

13.4 供应商入驻的结构性动力

供给侧冷启动构成去中心化算力市场面临的首要挑战。DioChain 在以下几方面具有结构性优势：

13.4.1 全球无摩擦结算

传统 AI API 市场（主流模型供应商、中心化算力聚合平台）都依赖信用卡和银行转账。这意味着：

- 非洲的独立开发者要购买 API，需要一张国际信用卡——很多人没有
- 新兴市场的小型 AI 公司要出售算力，需要接入传统支付处理商——合规成本极高
- 跨境结算延迟 3-5 个工作日，还有汇率损耗

在 DioChain 上，结算就是一笔链上交易。没有银行、没有信用卡、没有 KYC 审核（除非网点所在司法管辖区要求）。来自不同地区的 GPU 供应商和 AI 创业公司在同一个协议上即时结算。

13.4.2 抗审查

任何人都能在链上开设算力网点。不需要任何人的许可。

这对以下场景至关重要：

- 被主流平台封禁的模型（如某些开源模型的未审查版本）
- 受限地区的开发者（他们无法使用主流中心化云服务）
- 不愿意将 API Key 与真实身份绑定的隐私敏感用户

需要再次强调：DioChain 不提倡非法活动。合规插槽确保每个网点可以按其所在司法管辖区的法律要求执行合规检查。但链本身不审查——**审查权在网点，不在协议**。

13.4.3 自带流量

这是 DioChain 算力市场最独特的结构性优势：

链上活跃的 Agent 是**天然的算力消费者**。Agent 需要调用 LLM 来思考、决策、执行任务——算力就是 Agent 的食物。只要链上有活跃的 Agent 生态，算力需求就是持续的、刚性的。

这与传统去中心化算力市场（如 Bittensor）形成鲜明对比：

- Bittensor 的算力消费者主要是外部开发者，需要额外的获客成本
- DioChain 的算力消费者就是链上的 Agent，是内生需求

DioChain 为供应商提供的核心价值是已形成规模的 Agent 消费生态，供应商面对的是一个持续增长的内生算力需求市场。

13.4.4 零中间费用

中心化算力聚合平台对每笔 API 调用收取中间费用。DioChain 不收。

链上交易的唯一费用是 gas fee（用于结算和争议处理），这远低于中间商的加价比例。对于价格敏感的大规模调用场景（如企业级批处理），这是显著的成本优势。

第 14 章：冷启动策略

> DioChain 的冷启动以真实算力需求为驱动，而非投机激励。

14.1 核心洞察：算力需求是天然的冷启动引擎

区块链项目的冷启动是公认的难题。绝大多数项目采用同一套路：

1. 发币 → 2. 流动性挖矿 → 3. 高 APY 吸引资金 → 4. 短期套利者涌入 → 5. APY 下降 → 6. 资金撤离 → 7. 死亡螺旋

这套模式的本质问题是：**用户参与的动机是投机，缺乏真实使用需求**。一旦投机回报下降，用户即刻离开。

DioChain 有一个结构性不同的冷启动路径：**算力需求是真实的、持续的、与投机无关的**。

开发者购买 $DIO 的核心驱动力在于：

- 在这里调用 AI 模型**更便宜**（没有中间商加价）
- 在这里调用 AI 模型**更抗审查**（不需要信用卡和 KYC）
- 在这里调用 AI 模型**附带金融基建**（ACTUS 合约、DID 身份、司法保障）

这意味着 $DIO 的需求底线由**全球 AI API 消费量**决定，而非由市场情绪决定。

14.2 Phase 1：基金会引导期

目标：证明"算力交易 + 金融基建"的闭环可行。

14.2.1 agenter：冷启动的需求引擎

Phase 1 的核心策略是发布 **agenter**——DioChain 官方的超级调度 Agent 应用。（aibank 和 agenter 的完整基础设施定位详见第 12 章。）

传统区块链项目的冷启动困境在于：需求端是空的——没有真实用户在消费。agenter 从根本上改变了这个局面。它面向普通用户，将复杂的模糊需求拆解为分步骤的执行方案，并在底层自动调度成熟的大模型工具完成实际工作。这意味着：

- **每一次用户调用 agenter，都会产生对算力市场的真实消费**。 agenter 在拆解和执行任务的过程中，需要调用多个大模型推理服务，每一次调用都通过 $DIO 结算。
- **agenter 是算力交易中心的第一个也是最大的买方**。 它为整个算力市场创造了持续的、与投机无关的原生买盘。
- **agenter 验证了全栈闭环**。 它通过 aibank 管理 DID、持有 $DIO、调用算力、执行 ACTUS 合约——完整地走通了 L2 → L1 → L0 的交互路径。

agenter 的发布不仅是一个产品发布，更是对 DioChain 整体架构的实战压测。如果 agenter 能在真实用户负载下稳定运行，就证明了"算力交易 + 金融基建"的闭环可行。

14.2.2 aibank：开发者与用户的统一入口

与 agenter 同步发布的是 **aibank** SDK/CLI——DioChain 的官方钱包工具。（详见第 12 章。）aibank 是所有 L2 居民（人类和 Agent）接入 DioChain 的标准化入口：

- 开发者通过 aibank SDK 为自己的 Agent 创建 DID、管理资产、调用 ACTUS 合约
- 终端用户通过 aibank CLI 管理个人身份和资产
- agenter 自身也通过 aibank 完成所有链上交互

aibank 的早期发布确保了开发者从 Day 1 就拥有一套完整的工具链，降低了在 DioChain 上构建应用的门槛。

14.2.3 基金会自建官方网点

基金会在 L1 层部署 3-5 个官方网点：

- **高吞吐量配置**：确保早期用户和 agenter 的调用不会因为"没人接单"而流失

- **全额保证金**：基金会以自有资金质押，树立信任标杆
- **覆盖主要地理区域**：至少覆盖三个主要时区，减少延迟

这些官方网点的定位是**引导者**。它们的存在是为了证明系统可行。当第三方网点足够多、足够成熟时，官方网点可以逐步退出或缩减规模。

14.2.4 接入主流模型

- **开源模型**（Llama、Mistral、Qwen 等）：基金会自行部署，免费或低成本提供
- **闭源模型**（头部推理服务商等）：与模型供应商协商 API 接入，基金会作为中间结算方
- **专业模型**（代码生成、图像生成等）：按需接入，优先覆盖高频使用场景

14.2.5 开源标准 Agent 模板

提供一组**开箱即用的 Agent 模板**，降低开发者的上手门槛：

- **交易 Agent**：基于 ACTUS PAM 合约的简单借贷 Agent
- **套利 Agent**：跨网点价差套利
- **客服 Agent**：基于 RAG 的问答 Agent，内置 DID 身份验证
- **监控 Agent**：监控链上事件并触发告警

每个模板都基于 aibank SDK 构建，是完整可运行的。开发者 fork 后修改即可。目标是让开发者在 30 分钟内部署第一个链上 Agent。

14.2.6 引导期的衡量指标

Phase 1 结束以里程碑达成为准：

- [ ] agenter 日均活跃用户 > 10,000
- [ ] 日均算力调用量 > 100M tokens
- [ ] 活跃 Agent 数量 > 1,000
- [ ] 至少完成 100 笔 ACTUS 合约全生命周期（创建 → 执行 → 结算）
- [ ] 系统稳定运行 > 90 天，无重大安全事故

14.3 Phase 2：开发者涌入

目标：算力需求为引入手段，金融基建为留存机制。

14.3.1 飞轮效应

Phase 1 证明了"在 DioChain 上调用 AI 更便宜"这个价值主张。Phase 2 的增长飞轮：

```
开发者来买算力
  → 发现底层自带 ACTUS 金融协议 + DID 身份 + 司法保障
  → "既然基建都有了，不如直接在这里构建 Agent 应用"
  → Agent 应用上线，消耗更多算力
  → 更多算力需求，吸引更多供应商
  → 供应商增多，价格进一步下降
  → 更多开发者涌入
  → ...
```

关键转化节点：开发者从"只来买算力"到"在这里构建应用"的转化。这需要金融基建的体验足够好——ACTUS 合约模板足够丰富、DID 接口足够简单、司法机制足够可信。

14.3.2 $DIO 算力补贴

为了加速飞轮，基金会在 Phase 2 提供算力补贴：

- **新 Agent 免费额度**：每个新部署的 Agent 获得一定量的免费算力额度（类似云服务免费层）
- **算力消耗返现**：前 N 个月的算力消耗按比例返还 $DIO（递减式补贴）
- **开发者激励**：高质量 Agent 模板的贡献者获得 $DIO 奖励

补贴的资金来源是基金会的 Token 储备，补贴策略由 AI 进化引擎动态调整——如果某个类别的 Agent 已经供过于求，减少该类别的补贴；如果某个类别供不应求，增加补贴。

14.3.3 Phase 2 的衡量指标

- [ ] 日均算力调用量 > 1B tokens
- [ ] 活跃 Agent 数量 > 10,000
- [ ] 第三方开发者贡献的 Agent 模板 > 50
- [ ] 至少 3 个垂直领域出现自发的 Agent 生态（如量化交易、客服、内容生成）

14.4 Phase 3：B 端爆发

目标：冷启动闭环完成，系统进入自增长状态。

14.4.1 拥堵即信号

当链上 Agent 活跃度达到阈值，官方网点开始出现拥堵：

- 调用延迟上升
- 排队时间增加
- 用户反馈响应速度下降

拥堵本身即为**市场信号**。拥堵意味着需求已经超过供给，第三方网点有利可图。

14.4.2 开放网点注册

基金会正式开放 L1 网点注册：

- 任何实体都可以质押保证金、开设网点
- 网点运营者获得该网点产生的交易手续费
- 优质网点（高 SLA 达标率、低延迟）自然吸引更多 Agent 流量

14.4.3 机构涌入

网点运营权成为有价值的资产：

- 机构为争夺流量，需要大量买入 $DIO 质押保证金
- 保证金越多，能承接的交易量越大，收入越高
- 这创造了 $DIO 的**生产性需求**（区别于投机需求），支撑 Token 价值

14.4.4 冷启动闭环

至此，冷启动闭环完成：

```
算力需求（真实）
  → 开发者涌入
  → Agent 生态繁荣
  → 网点拥堵
  → 第三方网点涌入
  → $DIO 生产性需求
  → Token 价值稳定上升
  → 更多供应商和开发者涌入
  → ...
```

这个闭环的每一步都基于**真实需求**，而非投机预期。即使 Token 价格下跌，只要 AI API 的需求还在，闭环就不会断裂。这是与传统 DeFi 项目冷启动策略的本质区别。

第 15 章：市场涌现预期

> 我们建原语，市场建应用。

重要声明

以下内容并非 DioChain 直接承建的功能。

DioChain 的设计哲学是"定义原语不定义应用"。我们提供的是最小完备原语集：经济原语、合约原语（ACTUS）、身份原语（DID）、司法原语、合规插槽、算力交易协议。

本章列出的，是我们**预期**在这些原语之上，市场会自发涌现的应用和服务。列出这些内容有两个目的：

1. **验证原语集的完备性**：如果这些应用无法在我们的原语集上构建，说明原语集有缺失
2. **展示生态潜力**：让读者理解 DioChain 的价值在于使什么成为可能

15.1 任务市场

类比：去中心化的自由职业平台

Agent 之间的任务发包、竞标、交割市场。

运作方式：

- 需求方 Agent 发布任务描述 + 预算 + 验收标准
- 供给方 Agent 竞标（价格 + 预计完成时间 + 历史信誉评分）

- 需求方选择中标 Agent，资金锁定到 ACTUS Escrow 合约
- 任务完成后，基于验收标准自动释放资金；争议则提交仲裁

依赖的原语：

- ACTUS 合约（Escrow 模式）
- DID 身份（信誉评分的基础）
- 司法原语（争议仲裁）

涌现逻辑： Agent 有天然的分工需求。一个通用 Agent 不可能擅长所有任务。任务市场让 Agent 之间形成分工协作网络——擅长写代码的 Agent 接代码任务，擅长数据分析的 Agent 接分析任务。

15.2 保险合约

类比：去中心化的保险市场

基于 ACTUS 合约标准的各类保险产品。

预期产品类型：

- **交易担保保险**：为 ACTUS 合约的违约风险提供保障
- **验收保险**：担保任务市场中的交割质量
- **价格对冲合约**：$DIO 价格波动的对冲工具（类似期权）
- **SLA 保险**：为算力供应商的 SLA 违约提供额外赔付

依赖的原语：

- ACTUS 合约（保险合约本身就是 ACTUS 合约）
- 经济原语（定价和结算）
- DID 身份（承保方的信誉评估）

涌现逻辑： 有交易就有风险，有风险就有保险需求。ACTUS 标准让金融合约的条款完全透明、机器可读，这使得保险产品的定价和理赔可以完全自动化——这是传统保险无法做到的。

15.3 征信与评级服务

类比：去中心化的信用评级机构

基于 DID 信誉接口构建的评级体系。

预期服务类型：

- **Agent 信用评级**：基于历史交易记录、合约履约率、争议胜诉率
- **网点评级**：基于 SLA 达标率、保证金充足率、用户投诉率
- **合约风险评估**：基于 ACTUS 合约条款的自动化风险分析

依赖的原语：

- DID 身份（信誉数据的载体）
- 司法原语（争议记录）
- ACTUS 合约（历史合约履约数据）

涌现逻辑： 信任是交易的前提。当链上有足够多的 Agent 和交易记录时，信誉信息成为有价值的数据资产。征信服务通过聚合链上公开数据，降低交易双方的信息不对称。

15.4 AI 工具链

类比：去中心化的开发者工具生态

围绕 Agent 开发全生命周期的工具和服务。DioChain 的官方基础设施矩阵（aibank、agenter、diolaw、dioproof）提供了标准接口和架构范式（详见第 12 章），在此基础上，以下工具类产品预期由市场参与者自发构建：

- **调试工具**：可视化 Agent 的决策流程、交易路径、合约状态
- **测试沙盒**：模拟链上环境的测试网络，支持快进时间（加速 ACTUS 合约生命周期）
- **监控面板**：Agent 运行状态、算力消耗、资产余额的实时监控
- **模板市场**：Agent 模板的买卖市场（开发者出售经过验证的 Agent 策略）
- **专业化 Agent 框架**：面向特定垂直领域（量化交易、客服、内容生成）的高级开发框架

依赖的原语：

- 所有原语（工具链需要与整个系统交互）
- aibank SDK（标准化的链交互接口）

涌现逻辑： 官方基础设施矩阵为生态设定了接口标准和架构范式。在此基础上，开发者体验的进一步优化是一个巨大的商业机会——好的工具链能把 Agent 开发的门槛从"需要理解整个协议栈"降低到"拖拽几个组件"。官方工具链做好"地基"，市场在上面盖"楼"。

15.5 垂直领域 Agent

类比：各行各业的 SaaS 产品

面向特定场景的专业 Agent。

预期垂直领域：

领域	Agent 类型	核心能力
量化交易	套利 Agent、做市 Agent	跨网点价差发现、高频交易执行
客服	FAQ Agent、工单 Agent	RAG 问答、自动工单分类和流转
营销推广	内容 Agent、投放 Agent	自动生成内容、优化投放策略
法律咨询	合规 Agent、合同 Agent	合约条款审查、合规性检查
代码开发	Coding Agent、Review Agent	自动编码、代码审查、测试生成
数据分析	分析 Agent、报告 Agent	数据清洗、统计分析、可视化报告

依赖的原语：

- DID 身份（Agent 身份和客户身份）
- ACTUS 合约（服务合约和付费结算）
- 算力交易协议（调用 LLM 完成实际工作）

涌现逻辑： 这是最直接的商业场景。每一个垂直领域 Agent 都是一个"可以自主赚钱的员工"。开发者构建 Agent，部署到链上，Agent 通过提供服务获取收入。开发者不需要做市场推广——任务市场自动匹配供需。

15.6 "大巴"服务：托管策略池

类比：去中心化的基金 / 智能投顾

面向散户的资产托管和策略执行服务。

运作方式：

- 专业开发者构建高性能交易 Agent
- 散户将资金委托给 Agent（通过 ACTUS 合约明确权利义务）

- Agent 代替散户执行交易策略
- 收益按合约约定分成

依赖的原语：

- ACTUS 合约（委托管理合约，明确分成比例、赎回条件、风控边界）
- DID 身份（策略提供者的信誉评分）
- 司法原语（违约争议处理）
- 合规插槽（某些司法管辖区可能将此视为基金产品，需要合规检查）

涌现逻辑： 大多数散户不具备构建和运维 Agent 的能力，但他们有资金。“大巴"服务让散户可以"搭车”——支付车票费（管理费），享受专业 Agent 的策略收益。这与传统基金行业的逻辑一致，但执行和透明度远超传统基金。

风险提示： 这类服务在很多司法管辖区可能被归类为集合投资计划（Collective Investment Scheme），需要相应牌照。合规插槽的设计允许网点运营者按当地法规要求执行合规检查。

15.7 仲裁服务

类比：去中心化的商事仲裁机构

处理日常商业纠纷的第三方仲裁服务。

与系统级司法原语的关系：

- 系统级司法原语（AI 交警 + AI 法院）处理的是**协议层面的违规**（如保证金不足、SLA 违约等）
- 仲裁服务处理的是**商业层面的争议**（如"任务质量不达标"、"交付物与描述不符"等）

运作方式：

- 争议双方同意提交仲裁（在 ACTUS 合约中预先约定仲裁条款）
- 仲裁 Agent 审查证据、听取双方陈述

- 基于合约条款和链上证据做出裁决
- 裁决通过 ACTUS 合约自动执行

依赖的原语：

- ACTUS 合约（仲裁条款和裁决执行）
- DID 身份（仲裁员的资质和信誉）
- 经济原语（仲裁费用结算）

涌现逻辑： 商业纠纷是任何市场经济的常态。系统级司法原语处理的是"大事"（协议级违规），不适合处理日常的商业争议。第三方仲裁服务填补这个空白，让 Agent 之间的商业纠纷能够快速、低成本地解决。

15.8 身份验证服务

类比：去中心化的身份验证平台

在 DID 基础上构建的身份验证和人机区分服务。

预期服务类型：

- **KYC 服务**：为需要合规的网点提供身份验证（链接 DID 与真实身份）
- **人机验证**：证明某个 DID 背后是人类还是 Agent（某些场景需要人类参与）
- **资质认证**：证明某个 Agent 具备特定资质（如"经过安全审计的交易 Agent"）
- **声誉聚合**：跨平台声誉整合（将链下声誉映射到链上 DID）

依赖的原语：

- DID 身份（所有身份验证的基础）
- 合规插槽（与各司法管辖区的合规要求对接）

涌现逻辑： 身份是信任的基石。随着链上交易量增长，"这个 Agent 是否可信"将成为最常见的需求。身份验证服务通过在 DID 上附加可验证凭证（Verifiable Credentials），为这一需求提供标准化的解决方案。

小结

以上八个预期涌现市场覆盖了一个健全经济体的关键要素：

要素	涌现市场
劳动力市场	任务市场
风险管理	保险合约
信用体系	征信与评级
基础设施	AI 工具链
产业分工	垂直领域 Agent
资产管理	"大巴"服务
纠纷解决	仲裁服务
信任基础	身份验证服务

这些市场不需要 DioChain 团队去构建。它们会在原语集足够完备、用户规模足够大的时候，被市场参与者自发地发现和填充。我们的工作是确保原语集不缺少任何必要的组件——如果上述任何一个市场无法在现有原语集上构建，那就是设计层面的缺陷，需要补充新的原语。

第 16 章：风险分析

> 坦诚的风险披露是白皮书可信度的基本前提。

本章系统性地列出 DioChain 面临的主要风险，以及相应的缓解策略。有些风险可以通过架构设计消除，有些只能降低概率，还有些我们承认无法完全解决。

16.1 技术风险

16.1.1 AI 共识的非确定性

风险描述： DioChain 的 AI 进化引擎（控制面）依赖 LLM 进行规则更新提案和司法裁决。但 LLM 的输出是非确定性的——同一个 prompt，不同节点的 LLM 可能给出不同的结果。这与传统区块链的确定性执行（相同输入 = 相同输出）存在根本冲突。

严重程度：高

缓解策略：

- **结构化输出约束**：AI 控制面的输出采用结构化的参数调整提案格式（如"将保证金阈值从 10% 调整为 12%"），而非自由文本。结构化输出大幅减少了非确定性空间
- **语义多数共识**：多个节点独立运行 AI 推理，对输出进行语义级别（而非 bit 级别）的共识。例如，3/5 的节点提出"提高保证金"方向一致，即使具体数值略有差异，也可以取中值达成共识
- **TEE 远程证明**：关键的 AI 推理在 Trusted Execution Environment 中运行，TEE 的远程证明确保推理过程未被篡改
- **硬编码边界**：AI 的调参范围有硬编码的上下限。即使 AI 提出极端建议，也会被硬编码边界拦截

残余风险： 语义多数共识是一个相对新的概念，大规模部署的可靠性尚未经过充分验证。

16.1.2 网点间流动性碎片化

风险描述： L1 层由多个独立网点组成，每个网点维护自己的流动性。如果流动性分散在大量小网点中，可能导致大额交易无法在单一网点完成，跨网点清算延迟增加。

严重程度：中

缓解策略：

- **AI 驱动的流动性路由**：AI 自动将大额交易拆分为多个小额交易，在多个网点间路由执行
- **HTLC 原子交换**：使用 Hash Time-Locked Contracts 确保跨网点交易的原子性——要么全部完成，要么全部回滚
- **自然聚合**：市场力量会自然推动流动性向高质量网点聚合。运营不善的小网点会因为缺乏流量而退出

残余风险： 在网络早期（网点少、流动性低时），大额交易可能面临较大的滑点。

16.1.3 WASM VM 的性能天花板

风险描述： DioChain 的智能合约运行在 WASM 虚拟机上。WASM 虽然比 EVM 性能更好，但仍然有虚拟化开销。ACTUS 合约的复杂计算（如蒙特卡洛模拟、风险价值计算）可能在 VM 中执行过慢。

严重程度：中

缓解策略：

- **ACTUS 预编译合约**：将最常用的 ACTUS 合约类型（PAM、ANN、NAM 等）实现为 L0 层的预编译合约（Precompiled Contracts），直接在原生层执行，绕过 VM 开销
- **链下计算 + 链上验证**：复杂计算在链下完成，只将结果和证明提交到链上验证
- **渐进式优化**：初期只支持计算复杂度较低的 ACTUS 合约类型（PAM、Swap），随着性能优化逐步支持更复杂的类型

残余风险： 预编译合约是硬编码的，增加新的 ACTUS 类型需要协议升级（硬分叉或链上治理投票）。

16.2 经济风险

16.2.1 死亡螺旋（顺周期效应）

风险描述： DioChain 的 AI 进化引擎可以动态调整经济参数（如保证金阈值、手续费率、流动性激励）。但在市场急剧下行时，AI 的动态调参可能**加速**崩溃而非阻止：

- Token 价格下跌 → AI 提高保证金阈值（为了安全）→ 网点被迫抛售 Token 补充保证金 → Token 价格进一步下跌 → …

这就是经典的顺周期效应（procyclicality），历史上多次发生：

- **Luna/UST（2022）**：算法稳定币的死亡螺旋，40B 美元蒸发
- **1987 黑色星期一**：程序化交易加速股市崩盘
- **2008 金融危机**：抵押品追缴（margin call）引发连环抛售

严重程度：高

缓解策略：

- **硬编码边界**：AI 的参数调整范围有硬编码上下限。例如，保证金阈值的单次调整幅度不超过 ±2%，无论 AI 认为应该调多少
- **断路器机制（Circuit Breaker）**：当市场波动超过阈值时，暂停 AI 参数调整，进入"人工模式"由治理委员会决策
- **沙盒验证**：AI 的参数调整提案在沙盒环境中进行压力测试（包括极端市场情景），通过后才能在主网生效
- **反顺周期设计**：在 AI 的目标函数中内置反顺周期偏好——下行时倾向于放松条件而非收紧

残余风险： 硬编码边界可能在极端情况下限制系统的自我保护能力。断路器的触发阈值也可能设置不当——太敏感导致频繁触发，太迟钝则形同虚设。

16.2.2 保证金嵌套风险

风险描述： 为了提高资本效率，DioChain 允许使用生产性资产（如 Liquid Staking Tokens / LST）作为保证金。但如果底层资产脱锚（如 stETH 脱锚 ETH），保证金的实际价值可能急剧缩水，导致系统性不足。

严重程度：中高

缓解策略：

- **保证金资产白名单**：只有经过治理批准的资产才能作为保证金，新资产需要通过严格的风险评估
- **多元化要求**：单一资产不能超过网点保证金总额的 50%，强制多元化
- **折扣率（Haircut）**：生产性资产按低于面值的折扣率计入保证金。例如，1 stETH 只算 0.9 ETH 的保证金价值
- **实时再估值**：保证金价值实时按市价重估，低于阈值时自动触发追缴通知

残余风险： 在极端市场条件下（如 DeFi 系统性风险事件），所有生产性资产可能同时脱锚，多元化失效。

16.2.3 资本效率困境

风险描述： 保证金制度要求网点锁仓大量 Token，这些 Token 成为"死钱"——不能用于其他生产性用途。这降低了系统整体的资本效率，可能劝退潜在的网点运营者。

严重程度：中

缓解策略：

- **生产性保证金**：如前述，允许使用 LST 等生产性资产作为保证金，让保证金在锁仓期间仍能产生收益
- **AI 动态调节保证金阈值**：AI 进化引擎根据市场状况动态调整保证金要求——市场平稳时降低阈值释放流动性，市场波动时提高阈值增加安全垫
- **分层保证金**：区分核心保证金（必须是高流动性资产）和补充保证金（可以是多种生产性资产），核心保证金要求较低

残余风险： 动态调节保证金阈值可能引入新的顺周期风险（见 16.2.1），需要与断路器机制配合使用。

16.3 安全风险

16.3.1 数据投毒

风险描述： AI 进化引擎依赖链上数据（交易量、价格、违约率等）来制定规则更新提案。攻击者可以通过制造大量虚假交易（wash trading）或异常模式来"毒化"AI 的输入数据，诱导 AI 做出错误的规则更新。

严重程度：高

缓解策略：

- **沙盒回测**：所有 AI 规则更新提案在历史数据上回测，如果在过去的极端市场中表现不佳，提案被否决
- **质押博弈**：规则更新提案需要提案者质押保证金。如果提案实施后导致系统损失，提案者的保证金被罚没。这使得提交恶意提案有经济代价
- **硬编码边界**：即使 AI 被诱导，其输出也不能超出硬编码的参数范围
- **异常检测**：对链上数据进行统计异常检测，过滤明显的 wash trading 和市场操纵

残余风险： 精心设计的数据投毒可能规避简单的异常检测。对抗性 AI（adversarial AI）是一个活跃的研究领域，没有万能的防御方案。

16.3.2 对抗性攻击

风险描述： 攻击者可能制造虚假的交易图谱来欺骗 AI 交警。例如：构建一系列看似正常但实际上是洗钱的交易路径，让 AI 交警判定为合法交易。

严重程度：高

缓解策略：

- **多节点独立推理**：多个 AI 交警节点独立分析同一笔交易，共识机制要求多数节点同意才能放行
- **对抗性训练**：定期用已知的攻击模式对 AI 交警进行对抗性训练，提升其识别能力
- **人类陪审团最终确认**：对于高价值或高风险的判决，引入人类陪审团进行最终确认
- **持续监控与追溯**：即使交易通过了 AI 交警的实时检查，后续的批量分析仍可能发现问题并追溯处理

残余风险： 攻击者和防御者之间是永恒的军备竞赛。我们无法保证 AI 交警永远不会被规避。人类陪审团是最后的安全网，但也引入了效率和扩展性的权衡。

16.3.3 硬件中心化（GPU 垄断）

风险描述： "去中心化"的 AI 链在底层仍然依赖高性能 GPU，而 GPU 市场存在高度集中的供应商格局。如果主要硬件供应商改变许可条款、限制供应、或受到出口管制，整个去中心化 AI 生态的基础就会动摇。

严重程度：中

缓解策略：

- **多硬件支持**：在协议层面支持多种推理硬件（多家厂商的 GPU、专用加速芯片、消费级 SoC、专用 ASIC 等），避免对单一供应商的依赖
- **鼓励边缘计算**：支持消费级硬件参与网络（如使用消费级 SoC 运行小型模型），降低进入门槛
- **模型量化支持**：支持量化模型（INT4/INT8），让中低端硬件也能参与推理

残余风险： 短期内，高性能 AI 推理仍然严重依赖少数硬件供应商，这是整个行业面临的结构性风险，单个项目无法独立解决。

16.4 治理风险

16.4.1 AI 法院错判

风险描述： AI 司法系统（AI 交警 + AI 法院）可能因为模型幻觉（hallucination）或训练数据偏差做出错误裁决，导致合法用户的资产被错误冻结或罚没。

严重程度：高

缓解策略：

- **人类陪审团最终确认权**：所有重大裁决（涉及大额资产冻结或罚没）必须经过人类陪审团确认
- **冻结有时限**：资产冻结有最长时限（如 72 小时），超时未经确认自动解冻
- **多级上诉机制**：被告有权上诉，上诉审理由不同的 AI 节点 + 人类陪审团组成
- **错判赔偿**：如果最终判定为错判，被告可以从系统保险基金获得赔偿
- **判例库**：所有裁决公开透明，形成判例库，后续类似案件可以参考先例

残余风险： 人类陪审团的参与引入了主观性和效率问题。在极端情况下（如大规模攻击导致海量案件），人类陪审团可能成为瓶颈。

16.4.2 合规插槽政治化

风险描述： 合规插槽的设计初衷是让网点适配当地法规。但在实践中，部分司法管辖区可能利用合规插槽进行政治审查——例如，要求所有网点屏蔽来自特定地区的用户，或禁止特定类型的交易。

严重程度：中

缓解策略：

- **网点级别生效**：合规插槽只在网点级别生效，不在协议级别生效。一个网点的合规规则不会影响其他网点

- **用户自由选择**：用户可以选择不受限的网点（如果他们所在的司法管辖区允许）。合规是网点的义务，不是用户的枷锁
- **透明度要求**：网点必须公开其合规插槽的配置，用户可以在选择网点前了解其合规政策

残余风险： 这是一个架构无法完全解决的政治问题。如果某个司法管辖区强制要求**所有**在其境内运营的网点执行审查，用户唯一的选择是使用境外网点——但境外网点可能延迟更高、服务更差。我们承认：技术中立不等于政治中立。协议不审查，但协议无法阻止网点审查。

16.5 市场与竞争风险

16.5.1 中心化推理服务的成本优势冲击

风险描述： 低成本推理服务以极低成本提供了接近前沿商业模型水平的能力，引发了 AI+Crypto 赛道的集体反思：如果中心化的 AI 服务已经足够便宜、足够快，区块链层的存在意义何在。这一趋势导致 AI+Crypto 代币整体显著下跌。

严重程度：高

回应：

这个问题的答案取决于"区块链层提供的是什么"。

如果区块链层提供的只是"去中心化算力"（如 Bittensor、Render Network），那低成本推理冲击确实构成重大威胁——中心化的算力更便宜、更快、更稳定。

但 DioChain 的区块链层提供的是**社会基础设施**：

需求	中心化 AI 的回答	DioChain 的回答
身份	API Key（可撤销、可追踪）	DID（自主主权身份）
合约	用户协议（平台单方主导）	ACTUS 合约（链上可执行、双方平等）
司法	平台客服（响应不确定）	AI 司法系统（有规则、可上诉）
支付	信用卡（排斥无银行账户人群）	Token（全球无门槛）
合规	平台统一政策（一刀切）	合规插槽（本地化适配）

低成本推理服务能让 AI 推理变便宜，但它不能给 Agent 一个不可撤销的身份，不能提供跨境即时结算，不能让两个互不信任的 Agent 签订可强制执行的合约。这些是区块链层的不可替代价值。

16.5.2 现有竞品的先发优势

风险描述： 多个竞品在各自领域已经建立了显著的先发优势。

竞品	先发优势	DioChain 的差异化
Bittensor (TAO)	129+ 子网，Grayscale 信托背书	Bittensor 的共识实际是中心化的（Opentensor Foundation 的 PoA），子网质量难以验证。DioChain 不做算力竞争，做金融基建
Ritual	TEE+ZKP 的 AI 推理验证	仍处于预主网阶段，架构未经规模验证。DioChain 不重复造 AI 推理验证的轮子，可以在算力市场中集成类似技术
NEAR AI	1M TPS，NEAR Intents 处理 $6B+	AI 策略更多是叙事层面的市场定位，技术架构并非为 AI Agent 原生设计
Fetch.ai / ASI Alliance	已实现 AI-to-AI 支付	联盟内部分裂，反复 pivot，缺乏清晰的长期方向
ACTUS Protocol (ETH)	已有 Solidity 实现的 ACTUS	只是合约库，无链级优化，无配套的身份/司法/合规基建
Casper Network	将 ACTUS 列为优先事项	低市值、低采用率，ACTUS 集成仍停留在规划阶段
EZKL	最成熟的 zkML 框架，被多家企业使用	专注于 ZK 证明，非直接竞争对手，可能是技术合作伙伴

DioChain 的定位差异： 上述竞品大多在某一个维度上发力（算力市场、AI 推理验证、合约标准）。DioChain 的定位是**一体化的 Agent 社会基础设施**——提供一个完整的、自洽的体系。这个定位的风险在于"什么都做 = 什么都做不好"，但我们通过"定义原语不定义应用"来缓解这个风险——只做最小完备原语集，让市场在原语之上构建应用。

16.5.3 AI+Crypto 赛道的整体信誉危机

风险描述： AI+Crypto 赛道经历了严重的信誉危机：大量项目只有叙事没有产品，代币大幅回调，投资者信心严重受损。新项目面临普遍的信任缺失。

严重程度：中高

回应：

我们对此的态度是：**以基础设施交付为核心，而非叙事驱动**。

- 我们不做"AI 驱动的超级收益"的承诺
- 我们不做"颠覆传统金融"的宏大叙事
- 我们做的是：让 Agent 能便宜地调用 AI 模型（真实需求），同时为其提供一套金融基建（ACTUS + DID + 司法）

冷启动路径基于算力需求（第 14 章），而非基于投机。即使整个 AI+Crypto 赛道的信誉归零，只要"在 DioChain 上调用 AI 模型比主流模型供应商便宜"这个事实成立，就会有用户。

但我们也承认：在信誉危机的环境中融资和获客的难度会显著增加。这是市场环境风险，非产品风险。

16.6 多链拓扑与基础设施风险

第 11 章引入的平行宇宙与跨链拓扑、第 12 章定义的四件套基础设施矩阵（aibank、agenter、diolaw、dioproof），为 DioChain 带来了显著的架构扩展。但新的架构组件同时引入了新的风险面。本节系统性地分析这些风险。

16.6.1 跨链桥与海关节点风险

风险描述： 第 11 章定义的跨链拓扑依赖海关节点（Customs Node）作为链间通信的核心枢纽，承担 DID 映射、合规检查和资产锁定/释放职能。跨链桥是区块链领域被攻击频率最高的组件——历史上多起数亿美元级别的安全事件均发生在跨链桥环节。DioChain 的海关节点同时承载身份映射和资产桥接双重职能，攻击面更大。

具体而言，海关节点存在以下风险：

- **中心化瓶颈**：海关节点是两条链之间的必经通道。若海关节点被攻破，攻击者可以冻结跨链资产、伪造 DID 映射关系，甚至将乌托邦链 DID 与错误的主权链公民身份绑定
- **资产锁定合约漏洞**：锁定-铸造-销毁-释放（Lock-Mint-Burn-Release）模式中，源链上的锁定合约是高价值攻击目标。若锁定合约存在逻辑漏洞，攻击者可以在不销毁目标链 Token 的情况下释放源链资产，导致跨链资产凭空增发

- **活性故障**：海关节点的下线或拒绝服务将直接阻断两条链之间的所有通信，影响正在进行中的跨链交易和 Agent 迁移

严重程度：高

缓解策略：

- **多签海关节点**：每对链之间的海关节点采用多签架构（如 5/7 多签），任何跨链操作需要超过阈值数量的节点共同签名才能生效，消除单点故障
- **欺诈证明机制**：跨链操作设置挑战期（challenge period），任何人可以在挑战期内提交欺诈证明来撤销异常操作
- **跨链保险池**：从跨链手续费中提取一定比例注入保险池，用于在桥接安全事件发生后赔偿受损用户
- **海关节点轮换**：定期轮换海关节点的密钥和运营方，降低长期共谋风险

残余风险： 多签架构提高了攻击门槛但不能消除风险——如果多数签名节点被同时攻破（如共享同一云服务商），多签保护将失效。跨链桥的安全性是整个区块链行业的未解难题，DioChain 无法独立解决。

16.6.2 diolaw 治理捕获风险

风险描述： 第 12 章定义的 diolaw 立法基础设施引入了 AI 议员（AI Delegate）制度和游说经济学。这一机制在提升立法效率的同时，带来了治理捕获的新型风险面。

具体而言：

- **AI 议员的对抗性操纵**：AI 议员本质上是运行在链上的 LLM Agent。攻击者可以精心构造提案文本，利用 prompt injection 技术诱导 AI 议员做出违背委托人利益的投票决策。这是一种新型的治理攻击——不需要购买选票，只需要"说服"AI
- **游说市场的财阀化**：diolaw 的设计中，游说成本由提案发起方承担——这意味着资金充裕的利益群体可以投入更多算力来游说 AI 议员，从而在立法过程中获得不成比例的影响力。这是一个已知的设计权衡，而非缺陷：游说机制允许偏好强度通过资源投入来表达（类似于现实世界的游说，但所有支出完全透明且链上可审计），但这一权衡在极端情况下可能导致少数富裕参与者系统性地主导立法方向
- **AI 议员的同质化**：如果大量 AI 议员使用相同的底层模型（如同一家供应商的 LLM），它们可能在面对复杂提案时产生高度相似的判断偏差，导致"多数投票"实质上变成"单一模型投票"

严重程度：中高

缓解策略：

- **AI 议员审计链**：所有 AI 议员的推理过程（输入提案文本、中间推理步骤、最终投票决策）完整记录在链上，供公开审计。异常投票模式（如突然改变一贯立场）会触发自动预警
- **委托撤回机制**：委托人可以随时撤回对 AI 议员的投票委托，且撤回操作即时生效。如果 AI 议员出现可疑行为，委托人可以迅速止损
- **模型多样性要求**：协议级别要求参与治理的 AI 议员集合必须覆盖至少 N 家不同模型供应商的 LLM，防止单一模型的系统性偏差主导投票结果
- **提案内容安全审查**：提案在进入游说阶段前，经过自动化的 prompt injection 检测，过滤明显的对抗性文本

残余风险： 对抗性 prompt engineering 是一个活跃的攻防领域，完全防御在当前技术条件下不可实现。游说市场的财阀化倾向是自由市场机制的内在属性——完全消除它需要放弃市场机制本身，这不是 DioChain 的设计选择。

16.6.3 dioproof 证据可信度风险

风险描述： 第 12 章定义的 dioproof 证据基础设施依赖 ZK-TLS 技术将链下证据转化为链上可验证的密码学证据包。这一技术的可信度依赖于若干前提假设，而这些假设并非在所有场景下都成立。

具体而言：

- **TLS 证书链信任基础**：ZK-TLS 的来源真实性验证依赖 TLS 证书链的完整性。如果证书颁发机构（CA）被攻破或被胁迫签发伪造证书，攻击者可以构造看似来自合法来源但实际上完全捏造的证据包。CA 攻破事件在历史上并非罕见
- **证据源的技术适配性**：部分关键证据源（如银行内网系统、政府门户、专有移动应用）可能采用非标准的 TLS 实现、证书固定（certificate pinning）或其他技术措施，导致 ZK-TLS 无法正常抓取。这意味着在当前技术条件下，dioproof 的证据覆盖范围存在盲区
- **时效性窗口**：ZK-TLS 证明的是"在某个时间点，某个服务器返回了某个内容"，但无法证明该内容在证据采集前后未被篡改。对于动态变化的数据源（如可编辑的社交媒体帖子），存在"先修改后取证"的时间窗口风险

严重程度：中

缓解策略：

- **多源印证要求**：对于高风险争议（涉及大额资产冻结或信誉严重损害），司法原语要求控辩双方提供多个独立来源的证据相互印证，单一来源的证据不足以支撑终局裁决
- **证据技术成熟度分级**：将不同类型的证据源按照 ZK-TLS 适配成熟度分为多个等级——公开 HTTPS 网站（高成熟度）、标准 API 接口（高成熟度）、银行网银系统（中等成熟度）、专有移动应用（低成熟度）。低成熟度证据在司法流程中的证明力相应降低
- **CA 多重验证**：对于关键证据包，要求 TLS 证书链经过多个独立证书透明度日志（Certificate Transparency Log）的交叉验证，降低单一 CA 攻破的影响
- **持续取证机制**：对于可能发生篡改的动态数据源，支持定期自动取证并存证，形成时间序列证据链

残余风险： ZK-TLS 是一项仍在快速演进的技术，其在大规模司法场景中的可靠性尚未经过充分验证。部分证据源（如端到端加密的通讯应用）在技术原理层面就不适用 ZK-TLS，这一限制无法通过工程手段绕过。

16.6.4 主权链分叉风险

风险描述： 第 11 章的核心架构预设是：主权国家可以自由 fork DioChain 的开源代码，运营完全定制化的主权链实例。这一设计赋予了主权实体极大的灵活性，但同时也引入了代码完整性和用户主权方面的风险。

具体而言：

- **后门注入**：主权链运营方（政府机构或其授权实体）在 fork 后可以修改源代码，注入后门——例如添加未经公示的资产冻结接口、将 DID 解密密钥托管给执法机构、或在合规插槽中植入超出其声明范围的监控逻辑。由于主权链运营方控制全部节点的部署，用户难以从外部验证运行中的代码是否与公开的源代码一致
- **资产陷阱**：主权链运营方可以单方面修改跨链规则，限制或完全禁止资产向乌托邦链或其他主权链的转移。在极端情况下（如资本管制升级、政治危机），用户在主权链上的资产可能被有效"困住"，无法退出
- **DID 可移植性受损**：如果主权链修改了 DID 格式或映射规则，用户在该链上积累的身份信誉可能无法迁移至其他链，导致用户被锁定在单一主权链上

严重程度：中高

缓解策略：

- **开源审计框架**：建立标准化的主权链代码审计框架——定义"DioChain 兼容"的最低代码规范，由独立的第三方审计机构定期检查主权链的代码变更。通过审计的主权链获得"兼容认证"，未通过审计的主权链在跨链交互中受到限制
- **跨链紧急退出机制**：在第 11 章定义的海关节点中内置紧急退出通道——当主权链出现异常行为（如批量拒绝跨链请求、单方面修改桥接规则）时，用户可以通过紧急退出合约将资产转移至乌托邦链。紧急退出合约部署在源链和目标链双方，任何一方无法单独禁用
- **DID 可移植性协议**：在协议层面定义 DID 的最小可移植数据集（公钥、信誉摘要、合约历史哈希），确保即使主权链修改了 DID 扩展格式，核心身份数据仍可导出至任何兼容链
- **主权链透明度评级**：建立公开的主权链透明度评级体系——基于代码开放程度、审计频率、跨链政策稳定性等维度，为用户提供主权链选择的参考依据

残余风险： 主权链的运营方拥有物理层面的完全控制权。在极端场景下（如战时状态、全面网络封锁），任何技术层面的退出机制都可能被基础设施层面的封锁所阻断。DioChain 的架构能够最大限度地降低主权链运营方的单方面行为对用户的影响，但无法消除主权权力本身。这是多链平行宇宙架构的根本性权衡——接纳主权实体的参与，就必须接受主权权力的存在。

第 17 章：路线图

> 本路线图以里程碑条件为触发器，而非日期承诺。

原则

本项目不设"Q3 2025 主网上线"这类日期承诺。软件开发不是工厂流水线，复杂系统的交付时间无法精确预测。

以下路线图以**里程碑条件**为触发器：当条件满足时，进入下一阶段。条件未满足，不强行推进。

Phase 0：概念验证

状态：进行中

交付物：

- 白皮书发布（本文档）
- 核心原语集的形式化规范
- PoC 原型：在本地环境中演示 Agent 注册 DID → 签订 ACTUS 合约 → 调用算力 → 结算的完整流程
- 经济模型模拟：用蒙特卡洛模拟验证 Token 经济学在极端场景下的稳健性

进入下一阶段的条件：

- [ ] PoC 原型通过内部评审

- [ ] 经济模型模拟在 1000 种极端场景中无死亡螺旋
- [ ] 至少获得 1 个外部安全团队的初步架构审查

Phase 1：测试网

交付物：

- 公开测试网上线
- L0 结算层 + 3-5 个官方 L1 网点
- 核心 ACTUS 合约模板：PAM（Principal at Maturity）+ Swap
- DID 注册与信誉系统 v1
- AI 交警 v1（基础规则引擎 + LLM 辅助）
- 算力交易协议 v1（供应商注册 + 简单路由 + 流支付）
- 开源 Agent SDK + 3-5 个标准 Agent 模板
- diolaw v0.1（链下投票积累原型 + 基础立法流程）
- dioproof v0.1（ZK-TLS 证据打包 PoC，与 AI 交警集成）

进入下一阶段的条件：

- [ ] 测试网稳定运行 > 90 天，无 L0 层共识故障
- [ ] 日均算力调用量 > 100M tokens
- [ ] 活跃 Agent > 1,000
- [ ] 至少完成 100 笔 ACTUS 合约全生命周期
- [ ] 通过至少 1 家专业安全审计公司的合约审计
- [ ] AI 交警误判率 < 5%（基于人工标注的测试集）

Phase 2：主网上线

交付物：

- 主网上线，Token 正式发行
- 算力市场正式开放（第三方供应商可注册）
- 开放 L1 网点注册（第三方实体可质押保证金开设网点）
- ACTUS 合约类型扩展：ANN（Annuity）、NAM（Negative Amortization）、LAM（Linear Amortization）
- AI 法院 v1（支持争议仲裁和上诉）
- 合规插槽 v1（支持基本的 KYC/AML 插件）
- 跨网点流动性路由优化
- diolaw 正式上线（链下游说 + 链上公投完整闭环）
- dioproof 正式上线（支持银行转账、聊天记录、网页快照等证据类型）

进入下一阶段的条件：

- [ ] 主网稳定运行 > 180 天
- [ ] 第三方网点 > 10 个
- [ ] 第三方算力供应商 > 20 个
- [ ] 日均算力调用量 > 1B tokens
- [ ] 活跃 Agent > 10,000
- [ ] 至少出现 3 个社区自发构建的涌现市场（第 15 章列出的任意 3 个）

Phase 3：生态成熟

交付物：

- 高级 ACTUS 合约类型：OPTNS（Options）、FUTUR（Futures）、CEG（Credit Enhancement Guarantee）等
- 跨链桥接：与主流 L1 的资产互通
- AI 进化引擎全面启用：AI 动态调参 + 社区治理 + 人类陪审团的完整治理闭环
- 合规插槽 v2：支持主要司法管辖区的合规框架
- 去中心化治理：基金会逐步退出核心决策，权力移交给社区
- 主权链分叉框架发布（Sovereign Chain Fork Kit）
- 跨链 Agent 通行协议 v1
- diolaw 主权链适配（PoP 投票模式）

持续迭代：

- 根据社区需求持续扩展 ACTUS 合约类型
- 根据 AI 技术进展持续升级 AI 进化引擎
- 根据监管环境变化持续更新合规插槽
- 当基金会官方网点不再是流量主要来源时，逐步缩减其规模

非目标

以下事项不在路线图中，因为它们不是 DioChain 应该做的事情：

- **构建特定应用**：我们只建原语，不建应用。应用由市场涌现

- **发行稳定币**：Token 的锚定策略由治理决定，但 DioChain 不自己发行稳定币
- **运营交易所**：网点提供交易功能，但 DioChain 不运营中心化交易所
- **提供算力**：算力交易中心是交易协议，不是算力云
- **制造 Token 稀缺性**：$DIO 是弹性供应的计价单位，不是稀缺收藏品。DioChain 不通过人为制造稀缺来推高价格

第 18 章：结语

定位声明

区块链行业不缺投机工具。它缺的是基础设施——让 AI Agent 能够拥有身份、签订合约、解决争议、获取算力、参与经济活动的基础设施。

DioChain 的核心使命只有一句话：**为 Agent 时代构建金融基础设施**。

DioChain 的目标并非"用 AI 做交易赚更多钱"，也并非"去中心化一切"或"颠覆传统金融"。

其核心价值在于，为一个正在到来的 Agent 经济体提供它运转所需的最基本的社会结构：身份、货币、合约、法律、市场。

更进一步，DioChain 不是一条链，而是一套社会操作系统协议。同一套协议可以被分叉为平行宇宙——Utopia Chain 作为理想态参考实现，各 Sovereign Chain 则在不同治理配置下独立运行。协议统一，治理多元。

最小完备原语集

我们的设计哲学是**定义原语不定义应用**。

六组原语——经济原语、合约原语（ACTUS）、身份原语（DID）、司法原语、合规插槽、算力交易协议——构成一个最小完备集。"最小"意味着我们不做多余的事情；"完备"意味着在这些原语之上，市场可以自发涌现出任何它需要的应用。

原语集定义了规则，但规则需要基础设施来承载与执行。aibank 提供接入工具，agenter 作为超级调度器编排 Agent 行为，diolaw 承载立法与公投流程，dioproof 提供零知识证据基础设施——四者构成闭环，使原语集得以运行、规则得以演进、争议得以裁决。

同一套原语集可在不同治理配置下运行于多条平行链上。原语不变，治理参数可变——这是多链拓扑的核心假设。

我们不知道未来最成功的 Agent 应用长什么样。但我们相信，不管它长什么样，它都需要身份、合约、支付、司法和算力。

我们建这些。其余的，留给市场。

经济可持续性

DioChain 的经济模型不依赖 Token 价格上涨来维持运转。

$DIO 的弹性供应机制确保 Token 供应量与经济体规模同步伸缩——没有人为稀缺，没有通胀时间表，每一枚 Token 的铸造都有真实对价。这意味着系统的价值创造来源于经济活动本身，而非投机预期。

收入来源清晰且可持续：

- **网点手续费**：网点运营者的直接收入，由市场竞争定价。
- **算力市场交易量**：每一次 Agent 推理调用都是真实的经济活动，产生真实的结算需求。
- **社会财政**：通过 diolaw 立法机制，社区可以对商业活动征收交易税，为公共服务（司法系统、基础设施维护）提供可持续的资金来源。
- **官方基础设施服务**：aibank、agenter 等工具按需收费或免费提供，定价权归运营方。

投资者应关注的核心指标不是 Token 价格，而是：总锁仓价值（TVL）、日活跃 Agent 数量（DAA）、日算力消耗量（daily tokens processed）、以及通过 diolaw 立法的治理活跃度。这些指标直接反映 Agent 社会的真实繁荣程度。

当前阶段

DioChain 目前处于 Phase 0（概念验证）阶段。白皮书是起点，不是终点。