DioChain Whitepaper

Agent-First Digital Society Infrastructure for the Intelligent Era

Version: Draft v0.2

Table of Contents

Chapter 1: Summary

What is DioChain

DioChain is a set of on-chain social infrastructure for AI Agents. DioChain provides AI Agents with complete digital social infrastructure, covering identity, economy, law, and compliance.

Current blockchains are designed for human users—manual signing, manual transaction initiation, manual dispute resolution. The large-scale emergence of AI Agents is changing this premise: in the foreseeable future, the main actors in on-chain transactions will shift from humans to autonomously operating Agents. These Agents need far more than just a transfer channel; they need identity (DID), currency (Token), markets (buying and selling of compute power and services), law (dispute arbitration), compliance (meeting requirements of different jurisdictions)—in other words, they need a society.

Currently, no chain natively provides this complete set of social infrastructure.

Core Design

DioChain adopts a three-layer architecture:

- **L0 Central Settlement and Judicial Layer**: Provides final settlement, AI judicial arbitration, and global consensus.
- **L1 Distributed Branches Layer**: Margin-driven local clearinghouses that handle high-frequency transactions and periodically settle net amounts with L0.
- **L2 Agent and User Layer**: Agents and human users execute business logic here, accessing the network through L1.

Key technical decisions:

1. **AI does not execute in the hot path**. AI serves as an asynchronous evolution engine, continuously mining on-chain data and optimizing deterministic rule sets, rather than being embedded in the transaction execution path to introduce uncertainty.
2. **Native integration of the ACTUS financial contract standard**. All financial products share a set of machine-readable contract semantics, eliminating the need for Agents to individually interface with protocols.
3. **Compliance Slots**. Programmable compliance interfaces reserved for different jurisdictions, avoiding hard-coding a single legal system.
4. **Compute Trading Center**. Tokens can be directly used for pay-per-use calls to various AI models; compute power is the "food" for Agents, decoupled from monetary value.

For Whom It Is Built

The primary "residents" of DioChain are AI Agents—from simple trading bots to complex autonomous financial service providers. Human users exist as participants and principals, able to deploy, supervise, and benefit from Agent activities.

Targeted participants include: Agent developers needing autonomous economic identity; financial institutions requiring compliance frameworks; AI applications needing low-friction compute access; regulators seeking cross-jurisdictional interoperability; and ecosystem investors looking for long-term value in AI+blockchain infrastructure.

Why Now

In 2025, the AI+Crypto sector underwent a deep value reassessment: AI tokens overall fell by over 50%, with approximately $6B in market capitalization evaporated. The emergence of low-cost inference services proved a fundamental fact—if the blockchain layer does not provide irreplaceable specific functions, it is merely friction. The market has shifted from narrative-driven to value-driven.

Surviving projects share a common characteristic: the blockchain plays an irreplaceable role in them (incentive coordination, cryptographic verification, decentralized consensus). DioChain is designed based on this principle: the chain provides irreplaceable social primitives such as economy, law, identity, and compliance, while AI provides intelligence and adaptability, each fulfilling its own role.

Core Proposition

We only define primitives, not applications. Insurance, credit scoring, derivatives, risk control—these should emerge from the market, not be hard-coded at the protocol layer. Just as Ethereum provided EVM and ERC-20, and

Uniswap naturally emerged; DioChain provides a set of primitives for economy, contracts, identity, judiciary, and compliance, from which all applications will naturally emerge.

Existing AI+Blockchain projects each focus on a single functional module—better compute markets, better inference verification, better data trading. DioChain builds a complete system: a digital society where an Agent can fully live.

Chapter 2: The Era Context — The Infrastructure Gap in the Age of Intelligence

Blockchain and AI respectively address two fundamental types of problems in human society. However, their respective solutions come with profound structural compromises. Understanding these compromises is a prerequisite for understanding DioChain’s design decisions.

2.1 What Blockchain Solves and What It Compromises

Problems Solved

Blockchain uses cryptography and game theory to answer three ancient trust problems:

Byzantine Fault Tolerance (Byzantine Fault Tolerance). In an environment without a trusted central coordinator, how can a group of mutually distrusting participants reach consensus on a certain fact? The Nakamoto consensus provides an engineered answer: using Proof of Work to make the cost of forgi-

ng history higher than the benefit of honest participation. This solution is not theoretically perfect, but it has operated stably for over fifteen years in practice.

Double Spending (Double Spending). The essence of digital assets is infinitely replicable bit sequences. Blockchain ensures the same asset cannot be spent twice through a globally consensus-based ledger, achieving digital ownership for the first time without relying on intermediaries like banks.

Contract Execution (Code is Law). Smart contracts encode agreement terms into self-executing programs. Trading parties do not need to trust each other or rely on courts—they only need to trust the deterministic execution of the code. This holds real value in scenarios requiring frictionless, borderless collaboration.

The Price of Compromise

The cost of these achievements is equally profound but often underestimated amid industry enthusiasm:

Extreme Rigidity. The core of smart contracts is determinism—given the same input, they always produce the same output. This is the source of their trustworthiness and also their limitation. An `if-then` logic cannot handle fuzzy intentions ubiquitous in the real world, such as "roughly reasonable," "adjust based on market conditions," or "under controllable risk." Real-world contracts are full of gray areas, while smart contracts can only see black and white.

Adding more conditional branches cannot effectively cover such scenarios. Ambiguity is not an edge case; it is the norm in human economic activities. In traditional finance, loan terms can be renegotiated, insurance claims can

be arbitrated, and credit ratings can be manually adjusted—this flexibility is a crucial source of system resilience. Current smart contracts cannot provide this resilience.

The Contradiction Between Efficiency and Redundancy. To achieve trustless consensus, blockchain requires every node in the network to repeat the same computation. Thousands of nodes on the Ethereum mainnet independently execute every transaction—this is equivalent, in engineering terms, to the world's largest redundant computing network. Security comes from redundancy, but the cost of redundancy is extremely high computational expense and extremely low efficiency.

Solutions like L2 Rollup and sharding alleviate throughput bottlenecks but do not change the fundamental contradiction—the final confirmation of every transaction still relies on the global validation of the main chain.

Blindness to the External World. Blockchain is a closed deterministic system. It cannot perceive what happens in the off-chain world—today's gold price, whether a flight is delayed, or if a company defaults. Oracles attempt to bridge this gap, but they introduce a structural contradiction: a system whose core value proposition is "trustlessness" must trust a centralized or semi-centralized data source when acquiring external data.

The problem with Oracles is not just trust risk but also capability limits. Even if an Oracle is completely honest, it can only provide discrete, lagging data points, not a continuous understanding of the external world.

2.2 What AI Solves and What It Introduces

Problems Solved

AI (especially the latest advancements in large language models and multi-modal models) breaks through the capability boundaries of traditional computing in three dimensions:

Complex Pattern Recognition. Human financial analysts require years of training to identify patterns in noisy market data. AI models can capture high-dimensional patterns imperceptible to humans within spaces of tens of billions of parameters. In areas like fraud detection, credit assessment, and market anomaly identification, AI has already surpassed human expert performance in real-world deployments.

The Declining Cost Curve of Intellectual Labor. This is the most disruptive economic characteristic of AI. Traditional intellectual labor (legal review, financial analysis, compliance checks) costs scale linearly with volume—processing twice as many contracts requires twice as many lawyers. The marginal cost of AI approaches zero: the same model can simultaneously review contracts for a thousand clients, with the incremental cost being merely the electricity for inference computation. This represents a fundamental reshaping of cost structures.

Dynamic Adaptability. This is the core difference between AI and traditional rule engines. Rule engines fail when encountering unforeseen situations (or execute an inappropriate default branch). AI models can provide "roughly reasonable" outputs on inputs they have never seen before—precisely the capability smart contracts lack.

Problems Introduced

AI’s capabilities come with equally profound systemic risks:

Black Box Trust Crisis. A model with hundreds of billions of parameters makes a decision—rejecting a loan, flagging a transaction as fraudulent, assigning a credit score. No one can fully explain the causal chain behind this decision. In finance, unexplainable decisions pose both legal and ethical problems: regulations require financial institutions to be able to explain every decision made about consumers.

Research in Explainable AI (XAI) is progressing, but the explanations currently provided are closer to “post-hoc rationalization” than genuine reconstruction of the decision path. In high-risk financial decisions, this gap is unacceptable.

Extreme Monopoly of Power, Data, and Compute. Training cutting-edge AI models requires billions of dollars in capital investment and massive amounts of private data. This means the production of AI capabilities is structurally inclined toward extreme centralization—a few companies control the strongest models, the largest datasets, and the most computational resources. This creates a fundamental tension with blockchain’s value proposition of decentralization.

As of 2025, leading AI service providers occupy the vast majority of the global AI inference market. Mainstream open-source models and low-cost open-source inference models provide some balancing force, but the monopolistic landscape of training infrastructure has not fundamentally changed.

Difficulty Distinguishing Content Authenticity. AI-generated text, images, audio, and video have reached a level where humans cannot reliably distinguish authenticity. In finance, this means forged financial reports, fake

ction records, and counterfeit identity documents can be mass-produced at unprecedented low cost. Traditional defenses based on manual review are being bypassed.

2.3 Why Current AI+Blockchain Projects Fail

Market Data for 2025

2025 was a year of deep adjustment for the AI+Crypto sector. According to public market data:

- The overall market capitalization of AI-related tokens fell from a peak of approximately $12B to below approximately $6B, a drop of over 50%.
- This decline far exceeded the pullback of BTC and ETH during the same period, indicating the market selectively revalued the AI+Crypto narrative.
- Multiple once-hot projects (AI-based Meme tokens, AI trading signal platforms, AI NFT generators) went to zero or near zero.

This is not normal market cycle volatility; it is the market sending a clear signal: **narratives lacking actual functional support do not create sustainable value.**

The Impact of Low-Cost Inference Services

The release of low-cost open-source inference models in early 2025 was a turning point. It provided near-cutting-edge inference capabilities at a cost far lower than comparable products, directly exposing the core contradiction of many AI+Blockchain projects:

If centralized AI is faster, cheaper, and easier to access, what is the independent value of the blockchain layer in AI services?

This question posed a fundamental challenge to projects that "run AI models on-chain" or "use tokens to incentivize AI inference." Users do not care whether inference is "decentralized"—users care about the quality of results and the cost of access. When centralized solutions are superior in both dimensions, the blockchain layer becomes pure friction.

Common Characteristics of Surviving Projects

The projects that survived this shakeout share a clear common characteristic: **the blockchain plays an irreplaceable, concrete role in them.**

- **Bittensor (TAO)**: Coordinates contributors across 129+ subnets through token incentives. This large-scale, multi-party incentive coordination is difficult to achieve with purely centralized solutions. However, it only addresses the compute market and does not touch infrastructure like financial contracts, identity, or judicial systems.
- **Ritual**: Ensures the trustworthiness of AI inference results through cryptographic verification—this is the blockchain's irreplaceable function in this scenario. However, it targets general computation and lacks financial semantics.
- **NEAR AI**: Attempts to integrate AI into the chain's core layer but appears more as a narrative-level strategic adjustment, lacking structural product innovation.
- **Fetch.ai / ASI Alliance**: Early pioneers in the Agent economy, raising many correct questions, but alliance fragmentation and a long-term lack of breakthrough products have gradually sapped momentum.

Common Characteristics of Failed Projects

The pattern of failed projects is equally clear: **the blockchain layer only added friction without solving any problem that centralized solutions could not.**

Typical patterns include:

- "Use tokens to incentivize users to use our AI product"—token incentives are not a product. When subsidies stop, users return to better centralized alternatives.
- "Run AI inference on-chain"—100 times slower and 1000 times more expensive than centralized solutions, and unable to use the latest models.
- "AI-generated NFTs / AI-driven Meme tokens"—no sustainable value creation, purely speculative narratives.

Core lesson: **Blockchain must play an irreplaceable role in AI+Crypto applications. If the blockchain layer can be removed and the product still runs (or even runs better), then the project should not use blockchain.**

2.4 The Missing Piece: AI Agents Lack a Society to "Live" In

The above analysis points to a key structural gap:

Existing blockchains are designed for **human-manually signed transactions**. Every interaction assumes a human user operating a wallet, confirming transactions, and interpreting interfaces. This assumption is rapidly becoming invalid in the era of large-scale emergence of AI Agents.

An autonomously operating AI Agent requires the following infrastructure:

Identity (Identity)

An Agent needs a verifiable on-chain identity—a DID (Decentralized Identifier) bound to capabilities, historical records, and reputation, not an anonymous address. Without identity, interactions between Agents are

anonymous-to-anonymous, preventing trust establishment, reputation accumulation, and differentiation between a reliable service provider and a malicious attacker.

Currency (Currency)

An Agent needs a programmable medium of exchange to price and settle services. This requires going beyond simple ERC-20 tokens—it needs to support multiple anchoring mechanisms (stablecoins, carbon emission allowances, gold, etc.), enable micropayments between Agents (cent-level, millisecond-level), and seamlessly integrate with on-chain financial contract standards.

Marketplace (Marketplace)

Agents need to purchase compute power to sustain their "life"—calling AI models for inference is a prerequisite for Agents to perform any task. They also need to buy and sell services—an Agent skilled in risk assessment can provide credit rating services to other Agents. This requires a built-in, low-friction market infrastructure.

Law (Law)

When a dispute arises between two Agents—one fails to deliver services as per contract, another provides false data—there needs to be a dispute resolution mechanism. There is currently no native on-chain dispute resolution mechanism. Agents cannot resort to off-chain legal avenues or rely on human customer service. They need an on-chain judicial system: evidence preservation, emergency injunctions, automated arbitration.

Compliance (Compliance)

Agents operating across different jurisdictions need to comply with varying laws. Agents serving users in specific jurisdictions need to meet KYC/AML requirements; Agents serving users in certain jurisdictions need to comply with regional digital asset regulatory frameworks. Compliance cannot be an after-the-fact plugin; it needs to be a native capability at the infrastructure level.

High-Throughput, Low-Cost Transactions

Interaction frequency between Agents is far higher than between humans. A trading Agent might execute dozens of price queries and small transactions per second. The Gas models and throughput of current mainstream public chains are too costly and too slow for this usage pattern.

No Existing Chain Natively Provides This Complete Set of Social Infrastructure

- Ethereum provides smart contracts and DeFi primitives but lacks identity, judicial systems, compliance frameworks, and its Gas fees are unfriendly to Agents' high-frequency micropayments.
- Solana provides high throughput but lacks standardized financial contracts, identity systems, and compliance infrastructure.
- Cosmos provides cross-chain interoperability, but each application chain is isolated, lacking a unified economic and legal framework.
- Bittensor provides a compute market but lacks financial infrastructure, identity systems, and compliance frameworks.

These projects each solve a single-dimensional problem—faster consensus, better privacy, more efficient computation. But no project attempts to build a complete system—a digital society where AI Agents can survive and develop.

This is the gap DioChain aims to fill.

Chapter 3: The Vision of DioChain

3.1 An Agent-First Digital Society

In 2015, Ethereum defined itself as the "World Computer" – a universal, unstoppable computer shared by everyone. This positioning was accurate at the time: it expanded Bitcoin's capability boundary from "programmable money" to "programmable contracts."

DioChain's positioning is **"City-State for Agents"** – a digital space providing complete social infrastructure for AI Agents.

The key to this analogy lies in the word "society." A computer processes instructions; a city-state governs residents. A city-state needs far more than a faster processor – it requires a legal system, monetary system, identity registry, market rules, and law enforcement agencies. Similarly, Agents need far more than higher TPS – they need economic primitives, contract standards, identity systems, judicial arbitration, and compliance frameworks.

Furthermore, DioChain is not a closed city-state, but a set of city-state protocols that can be forked into multiple parallel universes. The officially operated "Utopia Chain" is a fully decentralized, AI-autonomous version; sove-

reign nations can fork the same codebase, bind identities to citizen IDs, replace AI judges with human courts, forming "Sovereign Chains." There is no absolute hierarchy between the two – we must respect an objective reality: governments command powerful resources, and their operational efficiency may far exceed that of the decentralized utopian version in many domains. Agents can traverse between multiple universes, much like citizens traveling between nations. The complete design of this multi-chain topology is detailed in Chapter 11.

From Organs to Organism

The evolutionary path of the blockchain industry over the past decade can be summarized as "organ optimization":

Dimension	Projects	Optimized "Organ"
Consensus Speed	Solana, Aptos, Sui	Faster Heart
Privacy Computing	Aztec, Penumbra	Better Immune System
Cross-Chain Interoperability	Cosmos IBC, LayerZero	More Flexible Joints
Compute Marketplace	Bittensor, Ritual	Bigger Muscles
ZK Proofs	zkSync, StarkNet	More Efficient Neural Conduction

Each project makes a specific "organ" stronger. But no project attempts to assemble these organs into a living organism.

The value of organs is real. But organs are not life. A powerful heart outside a body is just a muscle. An efficient compute marketplace, detached from identity systems, financial contract standards, and judicial frameworks, is merely a commodity exchange – offering no structural advantage over AWS Marketplace.

DioChain's core proposition is: **Only when economic, contract, identity, judicial, and compliance primitives are integrated into a unified framework can they produce emergent effects greater than the sum of their parts – just as organs form a body and give rise to "life."**

Finance itself is a social emergence. Loans require identity (who is borrowing?), credit (can they repay?), contracts (what are the terms?), law (what happens upon default?), and compliance (is it legal?). Scattering these functions across different chains, protocols, and bridges essentially forces an organic social process to run through spliced pipelines. What is lost is not just efficiency, but possibility – many financial products simply cannot exist in a spliced architecture.

3.2 Transportation Infrastructure Metaphor

To understand DioChain's system architecture more intuitively, we use a transportation infrastructure analogy. This analogy is a mental framework – it helps define the responsibility boundaries of each component.

Roads and Regulations

Chain = Roads + Regulations + Traffic Lights + Traffic Police

The chain provides infrastructure, not traffic itself. Roads do not decide where each vehicle goes, but they define the rules all vehicles must follow, the boundaries for movement, and the procedures for handling accidents.

- **Central Chain (L0) = Intercity Highways**. Low frequency, high security, final settlement. Intercity highways do not handle every intersection turn – they only manage high-level transit and key hubs between cities. L0 provides global

consensus, final settlement, and judicial arbitration, not participating in the per-transaction processing of high-frequency trades.

- **Distributed Branches (L1) = Urban Road Networks**. High frequency, low fees, local clearing. Urban roads handle over 90% of daily traffic. L1 branches are margin-driven local clearinghouses, processing high-frequency interactions between Agents, performing netting settlement with L0 periodically (not per transaction).

Vehicles

User-side AI = Vehicles

Agents are vehicles traveling on the roads. Different Agents have different "levels of autonomy":

Level	Analogy	Agent Type
L1 Assistance	Cruise Control	Simple conditional execution Bot (buy if price reaches X)
L2 Partial Automation	Lane Keeping + ACC	Strategy execution Agent (automatically trades per preset strategy)
L3 Conditional Automation	Highway Autopilot	Autonomous decision-making Agent (makes independent judgments and executes within authorized scope)
L4 High Automation	Urban Autopilot	Autonomous service Agent (independently operates financial services)
L5 Full Automation	Driverless	Fully autonomous Agent (independently decides business direction, manages assets, participates in governance)

The chain's design must support both simple L1 Bots and fully autonomous L5 Agents – just as city roads must accommodate both bicycles and autonomous trucks.

Traffic Regulations

ACTUS = Machine Traffic Regulations

In road traffic, all participants – whether human drivers or autonomous vehicles – must obey the same set of traffic regulations. This set of regulations is the common language for all traffic participants.

ACTUS (Algorithmic Contract Types Unified Standards) plays the same role in DioChain. It is the unified machine-readable standard for all financial contracts – defining 30+ standardized contract types (annuities, bonds, swaps, options, etc.), each with clear cash flow patterns and state transition rules.

This design is crucial because in an Agent-First economy, financial interactions between Agents must be machine-to-machine understandable. If each Agent uses its own invented contract format, the result is N^2 integration complexity – completely infeasible when the number of Agents reaches millions. ACTUS reduces this to N: each Agent only needs to implement the standard interface once.

Traffic Police and Courts

AI Judicial System = Traffic Police + Courts

A common misconception needs clarification here: DioChain's AI judicial system is **not a real-time traffic controller** – it does not make decisions on the execution path of each transaction. It is an **accident handling system**: intervening after problems occur, securing evidence, executing emergency injunctions, and making rulings.

Specifically:

- **Traffic Police Role (Real-time Monitoring + Emergency Injunction)**: AI continuously monitors on-chain activity patterns. When detecting system-level threats (large-scale abnormal withdrawals, contract exploit attacks, coordinated attacks, etc.), it can trigger emergency injunction measures – freezing suspected accounts, suspending suspicious contracts. This is similar to traffic police having the authority to immediately intercept dangerous driving without waiting for a court verdict.
- **Court Role (Post-facto Arbitration)**: When a contract dispute arises between two Agents, the AI arbitration system makes a ruling based on on-chain evidence (transaction records, contract states, communication history). The ruling result can be automatically executed (seizing margin, forced liquidation, etc.), without requiring the "voluntary cooperation" of the losing party.

The reason AI is not on the hot execution path is: AI reasoning results are probabilistic, while transaction settlement requires determinism. Embedding AI in the transaction execution path introduces unpredictability – precisely what blockchain aims to eliminate. The correct architecture is: **Deterministic systems handle routine affairs; AI systems handle anomalies and ambiguous scenarios.** Routine traffic relies on traffic lights and lane markings (deterministic rules); accidents and violations rely on traffic police and courts (AI judgment).

Special Lanes

ZK-Rollups / Dark Pools = Special Lanes

In a real transportation system, besides ordinary roads, there are special lanes like bus lanes, emergency lanes, and tunnels. They provide optimized passage for specific scenarios.

Special lanes in DioChain include:

- **ZK-Rollup Channels**: Provide batch compression verification for high-frequency trading scenarios, significantly reducing the cost of per-transaction on-chain recording. Similar to bus lanes – sacrificing some flexibility for efficiency in specific patterns.
- **Privacy Channels (Dark Pools)**: Provide zero-knowledge proof protection for large transactions requiring privacy. Similar to tunnels – external observers cannot see internal traffic, but the entrance and exit remain within the public road system, subject to rules.

These special lanes are optional infrastructure components, not required by all participants, but they are essential for specific scenarios.

3.3 Design Philosophy: Minimal Complete Primitives + Trust Market Emergence

We Only Provide Primitives

DioChain's design philosophy can be summarized in one sentence: **Define Primitives, not Applications.**

Primitives are the smallest, indivisible functional units. Ethereum's success largely stems from providing two key primitives – EVM (general computation environment) and ERC-20 (standardized token interface). Ethereum did not predefine what a DEX should look like, how a lending protocol should operate, or how a stablecoin should be pegged. It provided primitives, and the market gave rise to Uniswap, Aave, MakerDAO.

DioChain follows the same logic but provides primitives across broader dimensions:

Primitive Category	What it Provides	What it Does NOT Provide (Emerges from Market)
Economic Primitives	Configurably pegged Tokens, micropayment channels, netting settlement mechanisms	Specific stablecoin schemes, DeFi protocols, payment products
Contract Primitives	ACTUS standardized contract types, contract lifecycle management	Specific loan products, insurance products, derivatives
Identity Primitives	DID framework, Verifiable Credentials (VC), reputation scoring interface	Specific KYC service providers, credit scoring algorithms, credit rating methods
Judicial Primitives	Evidence fixation, emergency injunction, arbitration execution framework	Specific arbitration rule libraries, case law databases, mediation services
Compliance Primitives	Compliance Slots, regulatory reporting interface	Specific compliance solutions, anti-money laundering algorithms, tax calculations

Why Not Provide Applications?

This design reflects respect for the market's emergent capabilities.

A protocol-layer designer cannot foresee all application scenarios. The history of DeFi repeatedly proves this: Uniswap's constant product market maker, Compound's interest rate algorithm, MakerDAO's over-collateralized stablecoin – none of these innovations came from the Ethereum Foundation's design documents. They are optimal solutions that emerged after thousands of teams in the market experimented.

If Ethereum had hard-coded the "correct" DEX design at the protocol layer, Uniswap might never have appeared.

The same logic applies to DioChain. We do not know what the best insurance product in an Agent economy should look like. We do not know the optimal Agent credit scoring algorithm. We do not know which compliance solution best fits emerging markets. **What we know is: if the foundational primitives are sufficiently complete and flexible, the market will find the answers.**

Minimal Completeness

The words "minimal" and "complete" are equally important:

- **Minimal**: Each added primitive increases system complexity and attack surface. Primitives should be orthogonal (non-overlapping) and composable (freely combinable by upper-layer applications), not redundant.
- **Complete**: The primitive set must cover all fundamental dimensions required for a digital society to function – economy, contracts, identity, judiciary, compliance. Omitting any dimension forces the application layer to invent ad hoc solutions, which often become sources of system fragility.

Ethereum's primitive set (EVM + ERC-20) is minimal and complete for "programmable money," but incomplete for a "digital society" – it lacks identity, judicial, and compliance primitives. This is the root cause why DeFi remains constrained on compliance issues, relies on off-chain systems for identity verification, and lacks effective means for dispute resolution.

DioChain's primitive set aims for the minimal completeness of an "Agent digital society."

3.4 “Compute is Food, Token is Currency”

This section clarifies a common misconception about DioChain’s token economics.

Token is NOT a “Compute-Backed Currency”

In many AI+Blockchain projects, token value is pegged to compute power – 1 Token = X units of GPU compute. This design has a fundamental flaw: **AI inference costs are continuously and rapidly declining.**

From early large language models to current low-cost open-source inference models, the cost of the same quality inference output has dropped over 100-fold within two years. If the Token is pegged to compute, its purchasing power continuously depreciates as AI technology advances – holders are effectively shorting AI progress. This incentive structure is logically self-contradictory.

DioChain’s Token Design

DioChain’s Token is the **Unit of Account** for on-chain economic activity, not a representation of compute power.

The peg is configurable. The token’s pegging strategy depends on the choice of the deploying entity (chain operator or DAO):

- Peg to USD stablecoin: Suitable for deployment scenarios pursuing price stability.
- Peg to carbon emission allowances: Suitable for green finance scenarios.
- Peg to gold: Suitable for deployment scenarios pursuing anti-inflation properties.
- Floating exchange rate: Suitable for deployment scenarios pursuing market pricing.

This design embodies engineering pragmatism: "different deployment scenarios require different pegging strategies." A DioChain instance serving specific financial center institutions might peg to a local fiat stablecoin; an instance serving a carbon trading market might peg to carbon emission allowances. The protocol layer provides the framework for the pegging mechanism, not hard-coding a specific peg.

Compute as "Food"

In DioChain's economic model, compute power's role is **Agent "food,"** unrelated to currency pegging.

An AI Agent must "live" – i.e., run continuously, make decisions, provide services – it must consume compute power. Invoking AI models for inference is a prerequisite for an Agent to perform any task. In this sense, compute power is to Agents as food is to humans:

- Humans use currency to buy food to sustain life. Agents use Tokens to buy compute power to sustain "life."
- Declining food costs (agricultural technology progress) do not mean currency devaluation. Declining compute costs (AI technology progress) do not mean Token devaluation.
- Declining food costs mean the same income supports a better life. Declining compute costs mean the same Token enables Agents to perform more complex tasks.

DioChain's on-chain compute trading center allows Tokens to directly pay-per-use for calling various AI models (mainstream model providers, open-source models, etc.). Agents do not need to own GPUs – they purchase inference capability on-demand through the market, just as humans do not need to farm – they buy food from supermarkets.

The True Source of Value

Token value originates from **the sum of all economic activities in this digital society.**

A city's currency has value because all economic activities within the city – production, trade, services, taxation – are denominated and settled in that currency. Currency is the measure of economic activity.

DioChain Token's value logic is consistent:

- Every service transaction between Agents is settled in Token.
- Every financial contract's cash flows are denominated in Token.
- Every compute purchase consumes Token.
- Every branch settlement is conducted via Token.
- Every judicial arbitration's margin is deposited in Token.

When the scale of on-chain economic activity grows, the structural demand for Token grows. This is a sustainable value cycle driven by economic activity, not a value anchor tied to compute costs that would be eroded by technological progress.

> **Summary**: DioChain's vision is to build a city-state where Agents can live completely. We provide roads and regulations (infrastructure primitives), not manufacture vehicles (application-layer products). We enable Agents to use Tokens to buy food (compute power), provide services in the market (financial contracts), accumulate reputation (identity), operate under legal protection (judicial arbitration), and comply with local regulations (Compliance Slots). All specific applications and business models are left to emerge from the market.

{ “translated” : "# Chapter 4: System Topology — Three-Layer Architecture\n\n> **Core Argument**: DioChain adopts a three-layer separated architecture of L0/L1/L2. L0 provides finality and judicial authority, L1 eliminates credit risk at the code level through a margin mechanism, and L2 allows Agents to become "residents" of the chain rather than "components" of the chain. Each layer has its own responsibilities without intruding on others.\n\n—\n\n## 4.1 Global Architecture Diagram\n\n`mermaid\ngraph TB\n subgraph L2[\"L2 — Agent and User Layer (The Agentic Layer)\"]\n direction LR\n A1[\"Agent L3
Assisted Driving\"]\n A2[\"Agent L4
Highly Autonomous\"]\n A3[\"Agent L5
Fully Autonomous\"]\n U1[\"Human User
Principal/Strategy Setter\"]\n U1 -.→|Delegate| A2\n U1 -.→|Supervise| A1\n A3 -.→|Hire Humans| U1\n end\n\n subgraph L1[\"L1 — Distributed Branches Layer\"]\n direction LR\n B1[\"Primary Branch A
Margin 100M DIO\"]\n B2[\"Primary Branch B
Margin 50M DIO\"]\n B3[\"Secondary Branch A-1\"]\n B4[\"Secondary Branch A-2\"]\n B3 ⟶|Indirect Settlement| B1\n B4 -->|Indirect Settlement| B1\n B1 ⟷|HTLC Micro-Channel| B2\n end\n\n subgraph L0[\"L0 — Central Settlement and Judicial Layer (The Central Chain)\"]\n direction LR\n C1[\"Margin Pool\"]\n C2[\"Netting Settlement\"]\n C3[\"AI Judicial Engine\"]\n C4[\"Compliance Slots Registration\"]\n end\n\n A1 & A2 & A3 ⟶|Transaction/Call| L1\n B1 ⟶|Daily ZK-Proof Settlement| L0\n B2 ⟶|Daily ZK-Proof Settlement| L0\n\n style L0 fill:#1a1a2e,stroke:#e94560,color:#fff\n style L1 fill:#16213e,stroke:#0f3460,color:#fff\n style L2 fill:#0f3460,stroke:#533483,color:#fff\n` \n\n**Design Intent**: This three-layer structure directly maps to the real-world financial system—central settlement institutions (L0), clearing intermediaries (L1), and end-users (L2).

However, the difference is that L1 access is permissionless (anyone can open a branch by staking margin), and L2 residents are not only humans but also Agents. The three-layer architecture described in this chapter is the internal topology of a single DioChain instance. How multiple instances run in parallel under different governance frameworks and interconnect is detailed in Chapter 11.\n\n—\n\n## 4.2 L0 — Central Settlement and Judicial Layer (The Central Chain)\n\n### Scope of Responsibilities\n\nL0 is the "anchor" of the entire DioChain network. It only does four things:\n\n| Responsibility | Description | Frequency |\n|------|------|------|\n| Branch Margin Management | Lock/Release/Slash branch margin | Low frequency (branch lifecycle events) |\n| Daily Netting Settlement | Verify ZK-Proofs submitted by each branch, record ACTUS final state root | Once per settlement cycle |\n| AI Judicial Execution | Accept disputes, execute judgments, enforce liquidations | Very low frequency (triggered by disputes) |\n| Compliance Slots Registration | Register/update compliance rule modules for each jurisdiction | Very low frequency (when regulations change) |\n\n**Why not place more functions on L0?** Because the core contradiction in financial infrastructure is **security vs. throughput**. L0 chooses to completely sacrifice throughput in exchange for absolute security and finality. Any idea of "adding functions" to L0 dilutes this trade-off.\n\n### Consensus Mechanism: PoS + BFT\n\nL0 uses a variant of CometBFT (formerly Tendermint) as the consensus engine. Reasons for selection:\n\n**1. Instant Finality**\n\nFinancial settlement cannot tolerate "probabilistic confirmation." In the Bitcoin network, 6 block confirmations still have a (very small probability) rollback possibility. For a system that processes billions of DIO in net settlement daily, "very small probability" is unacceptable. BFT consensus guarantees: **once a block is committed, it cannot be rolled back**. This is a hard requirement for financial settlement.\n\n**2. Extremely Low TPS Demand**\n\nL0’ s daily load is extremely light:\n\n- Daily settlement: Assuming 10,000 primary branches, each submitting one settlement record = 10,000 TPS (instantaneous)\n- Judicial judgments: Possibly less than 100 per day on average\n- Margin operations: Possibly less than 1,000 per day

on average\n\nTotal daily transaction volume may not exceed 50,000. Setting block time to 10-30 seconds is sufficient. This means L0 can run on very conservative hardware, with extremely low operational costs for validators.\n\n**3. Validator Design**\n\n- Quantity: 100-300 validators\n- Staking threshold: High (specific value determined by DAO governance, initial suggestion 1,000,000 DIO)\n- Selection logic: Validator quantity does not pursue "many" but "stability." 100 validators with high stakes and high reputation better ensure the security of BFT consensus (BFT requires 2/3+ honest nodes; more validators increase coordination costs).\n\n### Storage Content\n\nL0 only stores the minimal necessary dataset on-chain:\n\n`text\nL0_Storage = {\n BranchRegistry: Map<BranchID, BranchMeta>, // Branch registry\n MarginPool: Map<BranchID, MarginBalance>, // Margin pool\n SettlementRoots: Map<Period, Map<BranchID, Root>>,// ACTUS state roots\n JudicialRecords: List<Judgment>, // Judicial judgments\n ComplianceSlots: Map<Jurisdiction, SlotConfig> // Compliance slots\n}\n` \n\n**Why not store transaction details?** Because transaction details belong to the internal data of L1 branches. L0 only cares about "whether your branch's final state in this settlement cycle is legal" (verified via ZK-Proof), not how many transactions you processed internally or the specific content of each transaction. This is similar to how the central settlement layer does not need to know every transfer detail of clearing intermediaries, only confirming that commercial banks' total accounts are balanced during daily clearing.\n\n—\n\n## 4.3 L1 — Distributed Branches Layer\n\n### Core Design: Margin-Driven Local Clearinghouse\n\nL1 is the most innovative layer in the DioChain architecture. Each branch is essentially a **margin-driven local clearinghouse**—a high-performance local transaction processor with margin hard-constraining risk exposure. It does not participate in global consensus (different from traditional nodes) and does not require independent blockchain security assumptions (different from sidechains).\n\n`mermaid\ngraph LR\n subgraph Branch[\"Branch Internal\"]\n ME[\"Matching Engine<br/>Million-level TPS\"]\n`

`SE[\"ACTUS State Machine\"]\n RM[\"Risk Management Module\"]\n ME ⟶ SE ⟶ RM\n end\n\n M[\"Margin<br/>Staked on L0\"] ⟶|Constraint| ME\n RM ⟶|Anomaly| ALERT[\"Circuit Breaker\"]\n SE ⟶|Daily| ZK[\"ZK-Proof Generator\"]\n ZK ⟶|Submit| L0[\"L0 Settlement\"]\n` \n\n### Transaction Limit = Margin Limit\n\nThis is DioChain's most important invariant:\n\n`text\ninvariant: branch.total-Exposure ⩽ branch.marginBalance\n` \n\n**At any moment, a branch's total risk exposure must not exceed its margin balance.** This rule is enforced as a hard constraint at the code level. If a branch stakes 1,000,000 DIO margin, it can only process up to 1,000,000 DIO in cumulative unsettled exposure. When exposure approaches the limit, new transactions are rejected; when exposure hits the limit, the circuit breaker automatically triggers.\n\n**Why design it this way?** Because the root cause of traditional financial crises is almost always "insolvency"—banks promise far more than they actually have. Fractional reserve systems, while improving capital efficiency, also plant the seeds of systemic risk. DioChain chooses another path: **completely eliminate the possibility of "insolvency" at the code level.** This means sacrificing some capital efficiency but gaining a system that will never collapse due to over-expansion of a branch.\n\n### Permissionless Access\n\nAny individual or organization that can stake sufficient margin can open a branch:\n\n1. Submit a registration request to L0's `BranchRegistry` contract\n2. Lock margin into the `MarginPool` \n3. Deploy branch software (open-source standardized stack)\n4. Start accepting transactions from Agents and users\n\n**No approval committee, no license, no KYC (at the branch level).** Margin is your "license"—it constrains risk more effectively than any permit.\n\n### Branch Internal Performance\n\nSince internal branch transactions do not require global consensus (only the branch's own bookkeeping, with ZK-Proof proving legality to L0 later), its performance limit approaches that of centralized servers:\n\n| Metric | Value | Comparison |\n|------|------|------|\n|

TPS | Million-level | Ethereum L1 ~15 TPS |\n| Latency | Sub-millisecond | Ethereum ~12 seconds |\n| Throughput Bottleneck | Memory/CPU | Consensus protocol |\n\n**This performance characteristic stems from the positive benefits of architectural decisions.** Removing consensus and settlement from the hot path leaves pure computational problems that can be directly solved by hardware capabilities.\n\n### Timed Netting Settlement\n\nBranches do not synchronize with L0 in real-time but settle in batches each settlement cycle (e.g., daily):\n\n `mermaid\nsequenceDiagram\n participant Branch as Branch\n participant ZKP as ZK-Proof Engine\n participant L0 as L0 Central Chain\n\n Note over Branch: During settlement cycle: high-speed transaction processing\n Branch->>Branch: Execute tens of thousands to millions of transactions\n Branch->>Branch: Maintain ACTUS contract state tree\n\n Note over Branch: Settlement time arrives\n Branch->>ZKP: Submit (initial state, final state, transaction set)\n ZKP->>ZKP: Generate ZK-Proof (STARK/SNARK)\n ZKP-->>Branch: proof\n Branch->>L0: Submit (final state root, proof)\n L0->>L0: Verify proof (millisecond-level)\n L0-->>Branch: Confirm/Reject\n` \n\n**Necessity of ZK-Proof**: L0 validators do not need to replay all transactions of a branch. When a branch processes 100 million transactions daily, requiring L0's 300 validators to each replay all transactions is computationally infeasible. ZK-Proof enables L0 to complete verification in milliseconds—proving the correctness of branch settlement results without knowing transaction details.\n\n**STARK vs SNARK Choice**: The initial phase recommends using STARK (no Trusted Setup, quantum-resistant), although proof size is larger. As SNARK schemes mature (e.g., Plonky2/Plonky3), switching is possible without changing the overall architecture. This is an implementation detail, not an architectural decision.\n\n### Secondary Branch System\n\nDioChain's branch system draws inspiration from the multi-layer clearing system of central banks and commercial banks:\n\n `mermaid\ngraph TB\n L0[\"L0 Central Chain\"]\n\n B1[\"Primary Branch A<br/>Margin: 100M DIO\"]\n B2[\"Primary Branch`

B
Margin: 50M DIO\"]\n\n S1[\"Secondary Branch A-1
Margin: 5M DIO\"]\n S2[\"Secondary Branch A-2
Margin: 3M DIO\"]\n S3[\"Secondary Branch B-1
Margin: 4M DIO\"]\n\n L0 ⟷|Direct Settlement| B1\n L0 ⟷|Direct Settlement| B2\n B1 ⟷|Indirect Settlement| S1\n B1 ⟷|Indirect Settlement| S2\n B2 ⟷|Indirect Settlement| S3\n\n\n**Why is a secondary system needed?**\n\n1. **Reduce L0 Load**: If 100,000 small branches all settle directly with L0, L0's validation burden would increase significantly. The secondary system lets L0 face only hundreds of primary branches.\n2. **Lower Entry Barrier**: Opening a primary branch may require 100 million DIO margin; but opening a secondary branch only requires staking a few million DIO to a primary branch. This allows small operators to participate.\n3. **Risk Layering**: Primary branches bear joint liability for their subordinate secondary branches. If a secondary branch has issues, the primary branch's margin compensates first. This creates natural supervision incentives—primary branches have motivation to audit their secondary branches' operations.\n\n**Risk Note**: The secondary branch system introduces additional settlement delays (secondary branches settle with primary branches first, then primary branches settle with L0). For time-sensitive large transactions, Agents should prioritize primary branches.\n\n### Inter-Branch Liquidity Routing\n\nWhen user Alice (at Branch A) transfers to user Bob (at Branch B), it does not need to go through L0:\n\nmermaid\nsequenceDiagram\n participant Alice as Alice @ Branch A\n participant BA as Branch A\n participant BB as Branch B\n participant Bob as Bob @ Branch B\n\n Alice-»BA: Transfer 100 DIO to Bob\n BA-»BA: Freeze 100 DIO from Alice\n BA-»BB: HTLC: 100 DIO, hash(secret), timeout=10min\n BB-»BB: Pre-credit Bob 100 DIO (conditionally locked)\n Bob-»BB: Confirm receipt\n BB-»BA: Reveal secret\n BA->>BA: Confirm, unfreeze completed\n Note over BA,BB: Unified netting during daily settlement\n\n\n**Key Mechanisms**:\n\n**1. Peer-to-Peer Micro-Channels**\n\nBranches with frequent interactions establish persistent

payment channels (similar to Lightning Network channels). Transactions within channels do not require on-chain confirmation, only synchronizing with L0 when channels close or during periodic settlement.\n\n**2. AI Prediction Engine Pre-Provisioning Liquidity**\n\nThis is where DioChain differs from simple payment channel networks: the AI Control Plane (detailed in Chapter 5) analyzes historical traffic patterns, predicts liquidity demand between branches, and pre-provisions funds in channels. For example, if AI detects "daily at 3 PM, transfer volume from Branch A to Branch B surges," it injects liquidity into channels in advance to avoid delays from opening temporary channels.\n\n**3. HTLC Atomic Swaps**\n\nCross-branch transfers use Hash Time-Locked Contracts (HTLC) to ensure atomicity—either both sides succeed or both roll back. There is no situation where "money is deducted from A but not received by B."\n\n**4. Daily Unified Netting**\n\nAll micro-channel net amounts are settled with L0 uniformly during daily settlement. This ensures micro-channels do not accumulate unsettled positions indefinitely.\n\n### Branch Economic Model\n\nBranches are not charities—they need a sustainable economic model:\n\n`text\nBranch Profit = Transaction fee income - Margin opportunity cost - Operational cost\n\nWhere:\n Transaction fee = Σ(each transaction × fee rate), fee rate determined by market competition\n Margin opportunity cost = Margin amount × risk-free interest rate\n Operational cost = Servers + Bandwidth + ZK-Proof generation computing power + Human resources\n` \n\n**Productive Use of Margin**: Margin does not have to be "dead money." Branches can use Liquid Staking Tokens (LST) as margin—for example, staking ETH to obtain stETH, then using stETH as margin. This way, margin still generates staking yield while locked.\n\n> **Risk Note**: Using LST as margin introduces nested risks (security of the LST protocol itself, de-pegging risk of LST). The system should set a discount rate (haircut) for LST margin, e.g., stETH margin valued at 95%. Specific discount rates are determined by DAO governance, with the AI Control Plane able to fine-tune within hard-coded bounds.\n\n**Unification of Compute Branches and Financial Branches**: A

branch can simultaneously provide financial clearing services and compute rental services (detailed in Chapter 13). These two businesses share the same margin and infrastructure, reducing marginal costs for operators. For Agents, "purchasing compute and financial services at the same branch" is the most natural user experience—no need for cross-branch scheduling.\n\n—\n\n## 4.4 L2 — Agent and User Layer (The Agentic Layer)\n\n### Positioning: Residents, Not Components\n\nL2 is not a layer of the blockchain—it is the "digital society layer" built on top of L0/L1. Agents and human users are "residents" of this society, using the financial infrastructure provided by L1 to live, work, and transact.\n\n**Why not make Agents part of the chain (e.g., smart contracts)?** Because Agent behavior is non-deterministic (relying on LLM reasoning), while blockchain core requirement is determinism (all nodes must produce the same output given the same input). Forcing non-deterministic logic into a deterministic system either sacrifices Agent capabilities or breaks chain consistency. The correct approach is to have Agents exist as "external users" of the chain, interacting through standardized interfaces.\n\n### Entry Point to the Digital Society: aibank\n\nDioChain is the underlying social infrastructure—L0 settlement layer, L1 branch layer, ACT

{ “translated” : "# Chapter 5: AI Evolution Engine — Control Plane and Execution Plane Separation\n\n> **Core Argument**: DioChain’s approach to AI is "control plane and execution plane separation"—inspired by the architectural philosophy of SDN (Software-Defined Networking). The execution plane is a purely deterministic, high-speed engine that processes each transaction; the control plane is an AI asynchronous daemon that continuously mines data and evolves rules. AI does not execute on the hot path; deterministic programs are prioritized for scenarios that can be resolved deterministically. This design fully leverages AI’s core strengths in pattern recognition and rule optimization.\n\n—\n\n## 5.1 Core Principle: Don’t Use AI for the Sake of Using AI\n\n### SDN Analogy\n\nIn the 2010s, the networking engineering field underwent a paradigm shift: SDN (Software-Defined Networking). The core idea is to separate the **control plane** from the **data plane** of network devices:\n\n- **Data Plane**: Switches/routers forward data packets at line rate according to forwarding tables. Pure hardware logic, deterministic, nanosecond-level latency.\n- **Control Plane**: The SDN Controller asynchronously collects network topology and traffic statistics, computes optimal routing strategies, and pushes forwarding tables to data plane devices. Software logic, capable of complex algorithms, but not on the hot path of data forwarding.\n\nThis separation brings two key benefits: the data plane can be optimized for extreme speed (since it doesn’t need to run complex logic), and the control plane can be optimized for extreme intelligence (since it’s not constrained by real-time requirements).\n\nDioChain’s architectural design for AI completely replicates this logic:\n\n`mermaid\ngraph LR\n subgraph CP[\"Control Plane\"]\n direction TB\n AI[\"AI Asynchronous Daemon\"]\n ML[\"Unsupervised Learning Engine\"]\n SB[\"Sandbox Backtesting Environment\"]\n AI ⟶ ML ⟶ SB\n end\n\n subgraph`

`DP[\"Execution Plane\"]\n direction TB\n RE[\"Rule Engine\"]\n AM[\"ACTUS State Machine\"]\n SM[\"Security Pattern Library\"]\n RE --> AM ⟶ SM\n end\n\n CP ⟶|\"Push Deterministic Rules
(Hot Update)\"| DP\n DP ⟶|\"Report Runtime Data
(Asynchronous)\"| CP\n\n style CP fill:#2d3436,stroke:#6c5ce7,color:#fff\n style DP fill:#2d3436,stroke:#00b894,color:#fff\n` \n\n### Three Iron Laws\n\n**Iron Law One: AI does not execute on the hot path.**\n\nThere are no AI inference calls on the processing path of any transaction. From entry into a branch to settlement completion, transactions follow a purely deterministic code path. The reason is simple: AI inference is probabilistic (the same input may yield different outputs), while financial transaction processing requires absolute determinism (all nodes must arrive at exactly the same result for the same transaction). Embedding probabilistic components into a deterministic pipeline is an architectural-level error.\n\n**Iron Law Two: Use deterministic programs for what can be solved deterministically.**\n\nIf a risk control rule can be written as `if exposure > threshold then reject`, then write it as deterministic code. There's no need for AI to "judge" whether a transaction should be rejected. AI's value lies not in executing known rules, but in **discovering unknown rules**—identifying patterns from massive data that human engineers haven't yet noticed and transforming them into new deterministic rules.\n\n**Iron Law Three: AI's output must be deterministic rules, not free text.**\n\nThe AI control plane does not output suggestions like "I think we should increase margin requirements." It outputs structured, directly deployable rule code:\n\n`json\n{\n \"rule_type\": \"margin_threshold_update\",\n \"target\": \"branch_category:tier_1\",\n \"parameter\": \"min_margin_ratio\",\n \"old_value\": 0.10,\n \"new_value\": 0.12,\n \"effective_after\": \"sandbox_validation\",\n \"evidence\": {\n \"data_window\": \"2025-01-01 to 2025-03-01\",\n \"anomaly_count\": 47,\n \"confidence\": 0.94\n }\n}\n` \n\nThis ensures AI's "creativity" is constrained within structured boundaries. The control plane can creatively discover new rules, but once

these rules enter the execution plane, they are deterministic—indistinguishable from rules handwritten by human engineers.\n\n—\n\n## 5.2 Execution Plane: Deterministic High-Speed Engine\n\n### Three Core Components\n\nThe execution plane consists of three deterministic components that collaboratively process each transaction:\n\n`mermaid\nsequenceDiagram\n participant TX as Transaction Request\n participant RE as Rule Engine\n participant AM as ACTUS State Machine\n participant SM as Security Pattern Library\n participant RES as Result\n\n TX->>RE: New transaction arrives\n RE->>RE: Match preset rules (limits/frequency/blacklist)\n alt Rule rejects\n RE-->>RES: Reject + reason code\n else Rule passes\n RE->>SM: Security pattern matching\n SM->>SM: Scan known attack patterns\n alt Matches attack pattern\n SM-->>RES: Reject + security alert\n else Security passes\n SM->>AM: Execute ACTUS state transition\n AM->>AM: Calculate cash flow / update contract state\n AM-->>RES: Success + new state\n end\n end\n` \n\n**1. Rule Engine**\n\nA collection of purely programmatic if-then-else rules. Each rule has explicit trigger conditions and execution actions:\n\n`text\nRule examples:\n- Single transaction limit: if tx.amount > agent.single_limit then REJECT\n- Daily cumulative limit: if agent.daily_total + tx.amount > agent.daily_limit then REJECT\n- Frequency limit: if agent.tx_count_last_minute > 100 then THROTTLE\n- Blacklist: if tx.counterparty in blacklist then REJECT + ALERT\n- Branch exposure: if branch.exposure + tx.amount > branch.margin then REJECT\n` \n\nThe execution complexity of the rule engine is O(n), where n is the number of active rules. By precompiling rules into decision trees or Aho-Corasick automata, matching time can be compressed to sub-microsecond levels.\n\n**2. ACTUS State Machine**\n\nState transition logic for all financial contracts (detailed in Chapter 7). This is a purely functional engine:\n\n`text\nnewState = actusTransition(currentState, event, contractTerms)\n` \n\nGiven the same current state, event, and contract terms,

the output is always the same. No randomness, no external dependencies (except market data provided by oracles, but oracle data itself is deterministic input).\n\n**3. Security Pattern Library**\n\nA continuously growing database of known attack patterns. Each pattern is a signature—describing the characteristics of a known malicious behavior:\n\n`text\nPattern examples:\n- Flash Loan Attack Pattern: Borrow large amount in same block → manipulate price → arbitrage → repay\n- Sandwich Attack Pattern: Insert transactions before and after target transaction\n- Reentrancy Pattern: Repeatedly trigger state changes in contract callback calls\n- Sybil Pattern: Many new identities initiate similar transactions in a short time\n` \n\nKey characteristics of the security pattern library: It is **append-only** (new patterns can only be added, existing patterns cannot be deleted), and each pattern has a clear hash identifier for auditing. Pattern growth comes from discoveries by the AI control plane—one of the control plane's most important outputs.\n\n### Performance Characteristics\n\nThe design goal of the execution plane is to make performance bottlenecks fall on hardware rather than software logic:\n\n| Component | Per-Transaction Processing Time | Bottleneck |\n|---|---|---|\n| Rule Engine | < 1 microsecond | CPU cache hit rate |\n| Security Pattern Matching | < 10 microseconds | Pattern library size |\n| ACTUS State Transition | < 100 microseconds | Contract complexity |\n| **Total** | **< 200 microseconds** | **Memory bandwidth** |\n\nComparison: A single LLM inference call typically has latency in the 100ms-10s range. If AI inference were placed on the hot path, the execution plane's throughput would drop by 3-5 orders of magnitude. This is the engineering rationale for "AI does not execute on the hot path."\n\n—\n\n## 5.3 Control Plane: AI Asynchronous Evolution\n\n### Architectural Positioning\n\nThe control plane is a set of asynchronous daemons running outside the execution plane. It does not process any real-time transactions; instead, it continuously consumes on-chain data streams, discovers patterns, and generates rule update proposals.\n\n`mermaid\ngraph TB\n subgraph DataSources[\"Data`

```
Sources\"]\n  MP[\"Mempool<br/>(Unconfirmed  Transaction  Pool)\"]\n
CT[\"Contract  Traces<br/>(ACTUS  State  Change  History)\"]\n
CS[\"Cross-Branch  State<br/>(Inter-branch  flow/balance/exposure)\"]\n
EX[\"External  Data<br/>(Market  prices/news/regulatory  changes)\"]\n
end\n\n  subgraph  Pipeline[\"AI  Pipeline\"]\n  direction  TB\n
FE[\"Feature  Engineering\"]\n  UL[\"Unsupervised  Learning\"]\n
RG[\"Rule  Generator\"]\n  FE  ⟶  UL  ⟶  RG\n  end\n\n  subgraph
Output[\"Output\"]\n  NR[\"New  Rule  Code\"]\n  PP[\"Parameter
Adjustment Proposals\"]\n SA[\"Security Alerts\"]\n end\n\n MP & CT &
CS  &  EX  ⟶  FE\n  RG  ⟶  NR  &  PP  &  SA\n\n  style  DataSources
fill:#2d3436,stroke:#636e72,color:#fff\n  style  Pipeline
fill:#2d3436,stroke:#6c5ce7,color:#fff\n  style  Output
fill:#2d3436,stroke:#fdcb6e,color:#fff\n
```

\n\n### Data Sources\n\n**1. Mempool (Unconfirmed Transaction Pool)**\n\nThe Mempool is a "collection of intentions"—transactions submitted but not yet executed. Analyzing the Mempool can:\n- Predict upcoming traffic spikes (sudden expansion of a branch’s Mempool)\n- Detect potential attacks (many structurally similar transactions appearing simultaneously)\n- Discover patterns of arbitrage opportunities (certain transaction sequences hint at MEV extraction)\n\n**2. Contract Traces**\n\nState change history of each ACTUS contract. Analyzing contract traces can:\n- Identify concentration risk of large contracts nearing maturity\n- Detect early signals of rising default rates\n- Detect abnormal contract parameter combinations (possible precursors to exploit attempts)\n\n**3. Cross-Branch State**\n\nBalance, exposure, and flow data of each branch. Analyzing cross-branch state can:\n- Predict changes in liquidity demand between branches\n- Detect abnormal fund flow patterns (money laundering characteristics)\n- Optimize liquidity routing strategies\n\n**4. External Data**\n\nOff-chain information accessed via oracles—market prices, macroeconomic indicators, regulatory policy changes, etc. This data helps AI

understand external drivers of on-chain behavior.\n\n### Learning Paradigm: Unsupervised First\n\nThe control plane’s learning paradigm prioritizes unsupervised learning. The reason is practical:\n\n**Labeled data is extremely scarce.** In a brand-new Agent economy, there is no historical default database, no labeled attack sample library, no expert-summarized "normal behavior baseline." Supervised learning requires large amounts of labeled data, which simply doesn’t exist in the system’s early stages.\n\nUnsupervised learning does not require labeled data. It discovers structure directly from raw data streams:\n\n| Technique | Role | Output |\n|----------|------|--------|\n| Anomaly Detection | Discover behaviors deviating from normal distribution | List of anomalous transactions + anomaly scores |\n| Clustering | Group similar behaviors | Behavior category labels |\n| Sequence Mining | Discover frequent patterns in transaction sequences | Behavior pattern signatures |\n| Graph Analysis | Discover anomalous structures in account relationship networks | Suspicious subgraphs |\n\n**Key trade-off**: Discoveries from unsupervised learning are "candidate signals," not "definitive conclusions." It produces false positives. This is why AI’s output must pass through a verification mechanism (see Section 5.4) before entering the execution plane—AI proposes hypotheses, the verification mechanism confirms or refutes them.\n\n### Output: Deterministic Rule Code\n\nThe final output of the AI control plane is **deterministic rule code that can be directly deployed to the execution plane**.\n\nThis conversion process occurs in three steps:\n\n`text\n1. AI discovers anomalous pattern\n → \"Over the past 7 days, 23 Agents initiated 10-15 transactions\n each with amounts of 999 DIO (just below the 1000 DIO reporting threshold)\n to the same branch between UTC 03:00-03:15 daily.\"\n\n2. AI generates rule hypothesis\n → \"This could be a structuring attack (splitting transactions to evade regulatory reporting).\"\n\n3. AI outputs deterministic rule code\n → {\n \"rule_id\": \"SEC-2025-0147\",\n \"type\": \"pattern_detection\",\n \"condition\": {\n \"time_window\": \"15min\",\n \"same_target_branch\": true,\n \"tx_count\": \"≥`

`5\",\n \"amount_range\": [900, 999],\n \"unique_agents\": \"≥ 3\"\n },\n \"action\": \"FLAG_FOR_REVIEW + RATE_LIMIT\",\n \"severity\": \"HIGH\"\n }\n` \n\nOnce this rule passes verification (see next section), it is hot-updated into the execution plane's rule engine. From then on, the execution plane matches this pattern at nanosecond speed—no need to call AI again.\n\n**Hot update mechanism**: Rule updates do not require restarting the execution plane. New rules are loaded into the rule engine's memory via atomic swap—the old rule set continues to serve current requests, and the new rule set takes effect at the next clock cycle after loading completes. This is similar to Nginx's hot reload mechanism.\n\n—\n\n## 5.4 Update Verification Mechanism\n\nAI is smart, but AI is also unreliable. A seemingly reasonable new rule could cause catastrophic consequences in edge cases. Therefore, every rule update from the AI control plane to the execution plane must pass through three verification gates.\n\n### First Gate: Sandbox Backtesting (Shadow Chain)\n\n`mermaid\ngraph LR\n NR[\"New Rule Proposal\"] ⟶ SC[\"Shadow Chain\"]\n HD[\"Historical Data Replay\"] ⟶ SC\n SC ⟶ R{\"Backtest Result\"}\n R ⟶|Pass| G2[\"Second Gate\"]\n R ⟶|Fail| RJ[\"Reject + Feedback to AI\"]\n\n style SC fill:#2d3436,stroke:#e17055,color:#fff\n` \n\nShadow Chain is a sandbox environment completely isolated from production, mirroring the full state of the execution plane. New rules undergo the following tests on Shadow Chain:\n\n**1. Historical Replay Test**\n\nReplay real transaction data from the past N days (default 30 days) on Shadow Chain, comparing behavior differences between old and new rules:\n\n`json\nBacktest Report = {\n total_transactions_replayed: 12,847,293,\n old_rule_rejects: 1,203,\n new_rule_rejects: 1,847,\n delta_rejects: +644, // New rule rejects 644 more transactions\n false_positive_estimate: 12, // About 12 of those are false rejections\n caught_known_attacks: 47, // Caught 47 known attacks\n missed_known_attacks: 0, // Did not miss any known attacks\n performance_impact: \"+0.3μs per tx\" // Impact on execution latency\n}\n` \n\n**2. Stress Test**\n\nInject extreme scenarios into

Shadow Chain (sudden 10x traffic, multiple branches simultaneously hitting margin limits, large-scale Agents initiating withdrawals simultaneously) and observe new rule performance under extreme conditions.\n\n**3. Adversarial Test**\n\nUse Red Team Agents to actively attempt to bypass the new rule. If the new rule is easily bypassed in adversarial testing, it's not robust enough and is returned to AI for regeneration.\n\n**Passing Conditions**:\n- False positive rate does not exceed 1.5x the baseline\n- No missed known attack patterns\n- Latency impact does not exceed 10 microseconds per transaction\n- Red Team successful bypass attempts < 5 out of 1000\n\n### Second Gate: Staking Game (Stake-and-Slash)\n\nSandbox backtesting verifies technical correctness. The staking game verifies economic incentive alignment.\n\n`mermaid\nsequenceDiagram\n participant P as Proposer\n participant V as Validator Community\n participant SC as Shadow Chain\n participant EF as Execution Plane\n\n P-->>P: Stake S tokens\n P-->>V: Submit rule update proposal\n Note over V: Challenge period (e.g., 48 hours)\n alt Someone challenges\n V-->>SC: Submit counterexample/adversarial evidence\n SC-->>SC: Verify counterexample\n alt Challenge succeeds\n SC-->>V: Reward challenger (from proposer's stake)\n SC-->>P: Slash part of stake\n else Challenge fails\n SC-->>P: Reward proposer (from challenger's stake)\n end\n else No challenges\n V-->>EF: Rule automatically takes effect\n Note over P: Stake returned after observation period\n end\n` \n\n**Mechanism Details**:\n\n- **Proposer Staking**: The AI control plane (or its operator) stakes a certain number of tokens for each rule update proposal. Staking amount is proportional to the rule's impact scope—rules affecting the entire network

Chapter 6: Economic Primitives

An Agent-First digital society requires native economic infrastructure—a unit of account, payment protocols, and a value circulation model. Traditional blockchain tokenomics often revolve around speculative narratives. DioChain’s Token is a **Unit of Account**, with its core function being to make economic activities between Agents measurable, settlable, and traceable.

Where are the boundaries? Economic primitives only define the underlying mechanisms for pricing, payment, and value circulation. Specific financial products (wealth management, lending, insurance), pricing strategies, and profit distribution schemes—these are all left to emerge from the market. The DioChain base protocol remains neutral, with no built-in fee extraction logic; fiscal policy is determined by the community through diolaw legislation.

6.1 Token Design

6.1.1 Unit of Account, Not a Speculative Asset

The primary identity of the DioChain Token is as a **Unit of Account**. It measures the value of all economic activities within the network: how many Tokens an Agent spends on purchasing compute power, what the margin for a contract is in Tokens, how many Tokens a branch is penalized.

This means:

- The design goal of the Token is **value stability**, not price appreciation.
- The Token is not an equity certificate; holding Tokens does not confer any governance rights or dividend rights in DioChain (governance rights are allocated by an independent governance mechanism).
- The Token discourages hoarding. Its optimal state is to **circulate rapidly** within the network, acting as a lubricant.

6.1.2 Pluggable Peg Framework

DioChain does not mandate what the Token must be pegged to—this is a deployment decision.

Different deployment scenarios (public chain, consortium chain, enterprise internal chain) face entirely different regulatory environments and business needs. DioChain provides a **Pluggable Peg Framework**, allowing deployers to choose a pegging strategy based on their requirements:

Pegging Strategy 1: Collateralized Debt Position (CDP)

Similar to MakerDAO’s CDP model:

- Users/institutions lock collateral (ETH, BTC, treasury tokens, etc.) into a collateral contract.
- The system mints Tokens based on a collateralization ratio (e.g., 150%).
- When the collateralization ratio falls below the liquidation threshold, automatic liquidation is triggered.
- **Applicable scenarios**: Crypto-native deployments where collateral is fully verifiable on-chain.

Pegging Strategy 2: Reserve-Backed

Similar to the USDC model:

- The issuer maintains an off-chain reserve pool (bank deposits, treasury bonds, etc.), backed 1:1.
- Regular third-party audits or Proof of Reserves are used to prove sufficient reserves.
- **Applicable scenarios**: Deployments by regulated financial institutions requiring compliance.

Pegging Strategy 3: Algorithmic

Similar to Frax's hybrid model:

- Partial collateralization + algorithmic supply adjustment.
- Tokens are minted when the price is above the peg and burned when below.
- **Applicable scenarios**: Scenarios pursuing capital efficiency, but requiring full understanding of de-pegging risks.

Whitepaper Example: Carbon Emission Peg

For illustrative purposes, this whitepaper uses a "carbon emission peg" as the default example in subsequent chapters:

- 1 Token = 1 ton of carbon emission equivalent in reduction credits.
- Real-time prices from carbon emission trading markets are accessed via oracles.
- An over-collateralized strategy is used, with certified carbon credit allowances as collateral.

Note: This is just an example. The framework also supports pegging to stablecoins (USD), gold, energy units, or even a basket of commodities. The choice of peg is a deployment decision, not a protocol constraint.

6.1.3 Compute Power is Not the Peg

A common misconception needs clarification: **Compute power is the "food" for Agents, not the peg for the Token.**

- Agents consume compute power to perform inference tasks, just as humans consume calories to sustain life.
- The price of compute power is determined by supply and demand in the compute power market (see Chapter 13), priced in Tokens.
- If compute power were used to peg the Token, the Token's value would fluctuate with compute power cost volatility (GPU price drops, new chip releases, etc.), contradicting the design goal of a "stable unit of account."

6.1.4 Core Uses of the Token

The Token has four core uses in the DioChain network:

Use	Description
Branch Margin	Opening a branch requires staking Tokens as margin to constrain branch behavior (see Chapter 4)
Transaction Fees	A small amount of Token is paid as a fee when submitting transactions to prevent spam attacks
Compute Power Purchase	Agents use Tokens to purchase inference compute power in the compute power market
Judicial Staking	Initiating judicial alerts, participating in juries, etc., require staking Tokens (see Chapter 9)

6.1.5 Elastic Supply

$DIO has no fixed total supply cap. The Token supply is determined by the actual demand of the economy—minted when users and Agents buy Tokens through the pegging mechanism (§ 6.1.2) and burned when they exit.

This fundamentally differs from Bitcoin's fixed-supply model or Ethereum's predetermined inflation model. DioChain's design logic is: **$DIO is a unit of account and medium of exchange, not a scarce coll-**

ectible. A healthy economy requires the money supply to expand and contract elastically with economic activity—artificially creating scarcity leads to a deflationary spiral, while excessive minting leads to value collapse.

Three constraints on elastic supply:

1. **Minting Requires Consideration**: Every $DIO minted must correspond to real collateral or reserves through the pegging framework (§ 6.1.2). Minting without consideration is impossible.
2. **Burning is Symmetric with Minting**: When users exit, Tokens are burned and collateral is released. The supply automatically contracts.
3. **Judicial Slashing Provides Additional Deflation**: The slashing and burning mechanism in § 6.3.2 acts as a "safety valve" under the elastic supply framework—even in extreme scenarios, penalties for malicious behavior automatically contract the supply.

This design means the "market cap" of $DIO is not a number to be "pumped," but a direct reflection of the economy's scale. The more prosperous the economy, the more $DIO circulates; if the economy shrinks, $DIO is automatically withdrawn. The core metrics for investors are not Token price, but total transaction volume, number of active Agents, and compute power consumption.

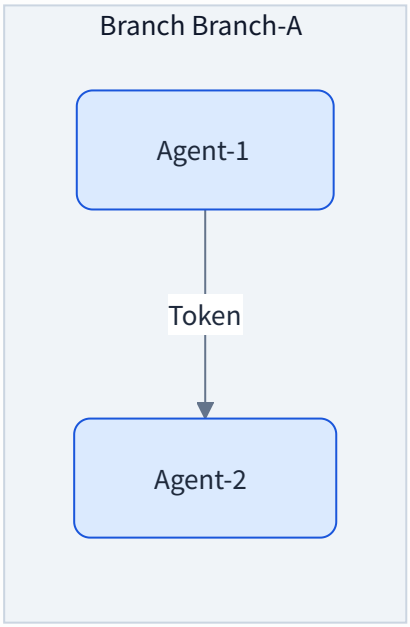
6.2 Payment Protocols

Economic interactions between Agents require multiple payment modes. DioChain provides three native payment primitives at the protocol layer, covering all scenarios from instant to long-term.

6.2.1 Instant Payment

Scenario: Agent A buys data from Agent B, with immediate payment and delivery.

- Within the same branch, instant payment confirmation delay is zero.
- The branch acts as a local clearinghouse, completing debits and credits directly on the local ledger.
- No need to wait for L0 global settlement—the branch's margin provides credit backing for these local transactions.
- Cross-branch instant payments are handled via liquidity channels between branches (similar to Lightning Network channels) or by waiting for the next L0 settlement cycle.



> Local ledger updates instantly with zero confirmation delay. Margin provides credit backing.

6.2.2 Streaming Payment

Scenario: An Agent calls an LLM for inference, billed in real-time based on the number of output tokens.

Streaming payment is the highest-frequency payment mode in the Agent economy. The core problem it solves is: **Compute power consumption is continuous and fine-grained, and should not be forced into discrete pre-paid packages.**

- The payer pre-locks a sum of Tokens into a streaming payment contract (similar to pre-authorization).
- The contract continuously deducts from the locked pool at an agreed rate (X Tokens per token, Y Tokens per second).
- Either party can terminate the stream at any time—payer stops consumption, payee stops service.
- Upon termination, the consumed portion goes to the payee, and the remainder is automatically refunded to the payer.
- Settlement granularity can reach micropayment levels (< 0.001 Token), as each micropayment does not require on-chain confirmation within a branch.

> Settled every 100ms, billed per token. Either party can close the channel at any time.

6.2.3 Escrow Payment

Scenario: Agent A delegates a task to Agent B, with payment only upon completion; or a conditional delivery transaction between two parties.

- The payer locks Tokens into an escrow contract.
- The escrow contract defines release conditions (time, multi-signature confirmation, oracle events, etc.).
- When conditions are met, Tokens are automatically released to the payee.
- If conditions are not met (timeout or dispute), Tokens are refunded to the payer or enter a dispute resolution process.

Escrow payment is the foundation for market-emergent infrastructure. Based on it, one can build:

- **Task Markets**: Agents post tasks, with rewards released upon acceptance.
- **Guaranteed Transactions**: When parties distrust each other, the escrow contract acts as a neutral third party.
- **Installment Payments**: Splitting a large payment into multiple milestones, released gradually.

DioChain only provides the escrow payment primitive. How acceptance criteria are defined and disputes are resolved—these are handled by upper-layer market contracts and arbitration contracts (market-emergent).

6.3 Value Sustainability Model

A healthy economic system requires a sustainable value cycle. DioChain's value model is designed around four principles: **Revenue, Burning, Productivity, and Protocol Neutrality.**

6.3.1 Revenue: Branch Transaction Fees

- Each transaction within a branch incurs a small fee.
- Fees belong to the branch operator as direct compensation for operating infrastructure and providing clearing services.
- Fee rates are set by the branch, with market competition driving them to reasonable levels—if rates are too high, Agents migrate to other branches.

6.3.2 Burning: Deflation Mechanism via Judicial Slashing

- When a branch is deemed malicious by judicial primitives (see Chapter 9), its margin is slashed.
- Distribution of slashed margin:
  - Part compensates victims.
  - Part is **permanently burned**, removed from circulation.
- The burning mechanism provides mild deflationary pressure on the Token, offsetting potential over-minting.
- Note: The burning mechanism arises as a byproduct of judicial enforcement and should not be artificially expanded to create deflation.

6.3.3 Productive Margin

A branch’ s margin is a large sum of capital; if idle, it represents significant waste. DioChain allows margin to be used as **productive assets**:

- Branches can deposit margin into productive protocols to earn yields (similar to Liquid Staking Tokens, LST).
- As long as the equivalent value of the margin remains above the minimum requirement.

Risks that must be noted:

- **Nested Risk**: If the margin is in LST and the underlying protocol fails (de-pegging, attack), the actual value of the margin may fall below the minimum requirement, putting the branch in an undercollateralized state.
- **Mitigation Measures**: The protocol layer should define an allowed whitelist of productive assets, maximum nesting levels, and real-time monitoring and alert mechanisms for margin value.

6.3.4 Protocol Neutrality and Social Treasury

The DioChain base protocol does not extract commissions from transactions. This is a core design decision at the protocol layer—the protocol is infrastructure, not a rent-seeker.

But DioChain is not just a protocol; it is a social operating system. Any society needs fiscal revenue to maintain public services—operation of the judicial system, infrastructure maintenance, public goods development. DioChain delegates fiscal policy decisions to the community through diolaw legislation (Chapter 12):

- **Transaction Taxes**: The community can legislate via diolaw to impose transaction taxes on specific types of commercial activities. Tax rates, scope, and exemptions are all determined through the legislative process.

- **Infrastructure Usage Fees**: Official infrastructure (aibank, agenter, etc.) can charge fees as needed or be provided free—pricing strategies are decided by operators based on market conditions.
- **Branch Transaction Fees**: Branch operators' autonomous pricing remains unchanged (§ 6.3.1).

Distinguishing Protocol Layer and Social Layer: The key distinction here is—the protocol layer (L0/L1 consensus and settlement mechanisms) has no built-in extraction logic; the social layer (via diolaw legislation) can establish fiscal systems on top of the protocol. This is isomorphic to human society: TCP/IP protocols do not charge fees, but governments can tax internet commerce. The protocol remains neutral, society decides fiscal policy.

Tiered Governance: Modifications to core economic parameters (e.g., transaction tax caps, minting rules) require higher governance thresholds—not only passing diolaw's standard legislative process but also involving foundation members holding specific governance NFTs in deliberation. This prevents rash changes to the economic foundation while retaining evolution possibilities.

6.4 Token Distribution Framework

Since $DIO uses an elastic supply mechanism (§ 6.1.5), there is no "total distribution" in the traditional sense. There is no pre-mining, no team lockups, no investor allocations—every Token is minted on-demand through the pegging mechanism.

The core issue of Token distribution thus shifts from "who gets how much" to "how is minting authority governed":

- **Minting Authority**: Automatically executed by the pegging framework’ s smart contracts; any entity meeting collateral conditions can mint.
- **Initial Liquidity**: In Phase 0-1, the foundation provides initial liquidity via a reserve-backed pegging strategy, ensuring early participants can acquire $DIO.
- **Long-term Evolution**: The choice and parameter adjustments of pegging strategies are decided through the diolaw legislative process. Different chain instances (Utopia Chain vs. Sovereign Chain) can adopt different pegging strategies.

Specific initial liquidity scales and pegging parameters will be determined during Phase 1 testnet based on economic model simulations.

Chapter 7: Contract Primitives and the ACTUS Standard

Economic activities between AI Agents are far more complex than simple "transfers"—they include lending, swaps, options, insurance, and structured products. The traditional DeFi approach is to have each team handwrite these financial logics from scratch in Solidity, resulting in: vastly different code, bugs everywhere, zero interoperability, and unmeasurable systemic risk. DioChain natively integrates the ACTUS financial contract standard at the virtual machine level, allowing all financial activities to run on a standardized state machine.

Where are the boundaries? Contract primitives define the "execution engine for financial contracts" and the "runtime environment for generic contracts." As for what specific financial products to deploy, how to set parameters, and how to manage risk—these are decisions for market participants, not responsibilities of the protocol layer.

7.1 Native Integration of the ACTUS Financial Contract Standard

7.1.1 What is ACTUS

ACTUS (Algorithmic Contract Types Unified Standards) is an open standard that **reduces all financial contracts to standardized state machines and cash flow events**.

Its core insight is: no matter how complex a financial contract may be in legal text, its underlying economic substance can be decomposed into **a series of cash flow events triggered at specific times by specific events**. ACTUS categorizes these patterns into about 30 standard contract types, each with a strict mathematical definition:

Contract Type	Abbreviation	Description
Principal at Maturity	PAM	Repay principal at maturity, pay interest periodically (most basic fixed-rate loan)
Annuity	ANN	Annuity, equal installments of principal and interest
Linear Amortizer	LAM	Linear amortization
Swap	SWPP	Interest rate swap, currency swap
Option	OPTNS	European/American option
Collateral	CEC/CEG	Collateral management
…	…	About 30 types in total, covering most financial scenarios

The cash flow calculation for each contract type is **completely deterministic and mathematically precise**. Given the same parameters (principal, interest rate, term, calendar rules, etc.) and the same market state (interest rate curve, exchange rate), any system implementing the ACTUS standard must produce identical results.

7.1.2 Why Choose ACTUS

In the context of DioChain, the value of ACTUS lies not only in “standardization” but also in its deep alignment with the Agent-First architecture:

1. A Common “Financial Language” for All AI Agents

When millions of Agents participate in financial activities within the network, if each Agent uses custom black-box code for financial contracts, interoperability becomes extremely difficult. ACTUS provides a unified vocabulary and grammar—any Agent can unambiguously read, parse, and evaluate the state and future cash flows of any ACTUS contract.

2. 100% Machine-Readable and Computable Network-Wide Financial State for AI

This is ACTUS’s most powerful attribute. Because all financial contracts follow a standard state machine, the AI Evolution Engine (see Chapter 5) can:

- Traverse all active contracts across the entire network.
- Precisely calculate the cash flows of each contract at any future time point.
- Perform network-wide stress tests (Monte Carlo simulations) under extreme scenarios.
- Discover hidden systemic risk linkages (e.g., which contract liquidations would be triggered if asset X drops 30%, and which assets would further decline due to those liquidations).

If financial contracts were non-standard code handwritten in Solidity, the above analyses would be nearly impossible to implement technically.

3. Systemic Risk Can Be Simulated and Predicted

The common lesson from the 2008 financial crisis and the 2022 Terra/Luna collapse is: when financial products are nested layer upon layer without a unified descriptive standard, systemic collapse is difficult to foresee. ACTUS turns each contract into a computable mathematical object, making quantitative analysis of network-wide risk possible.

4. Already Recognized by Authorities

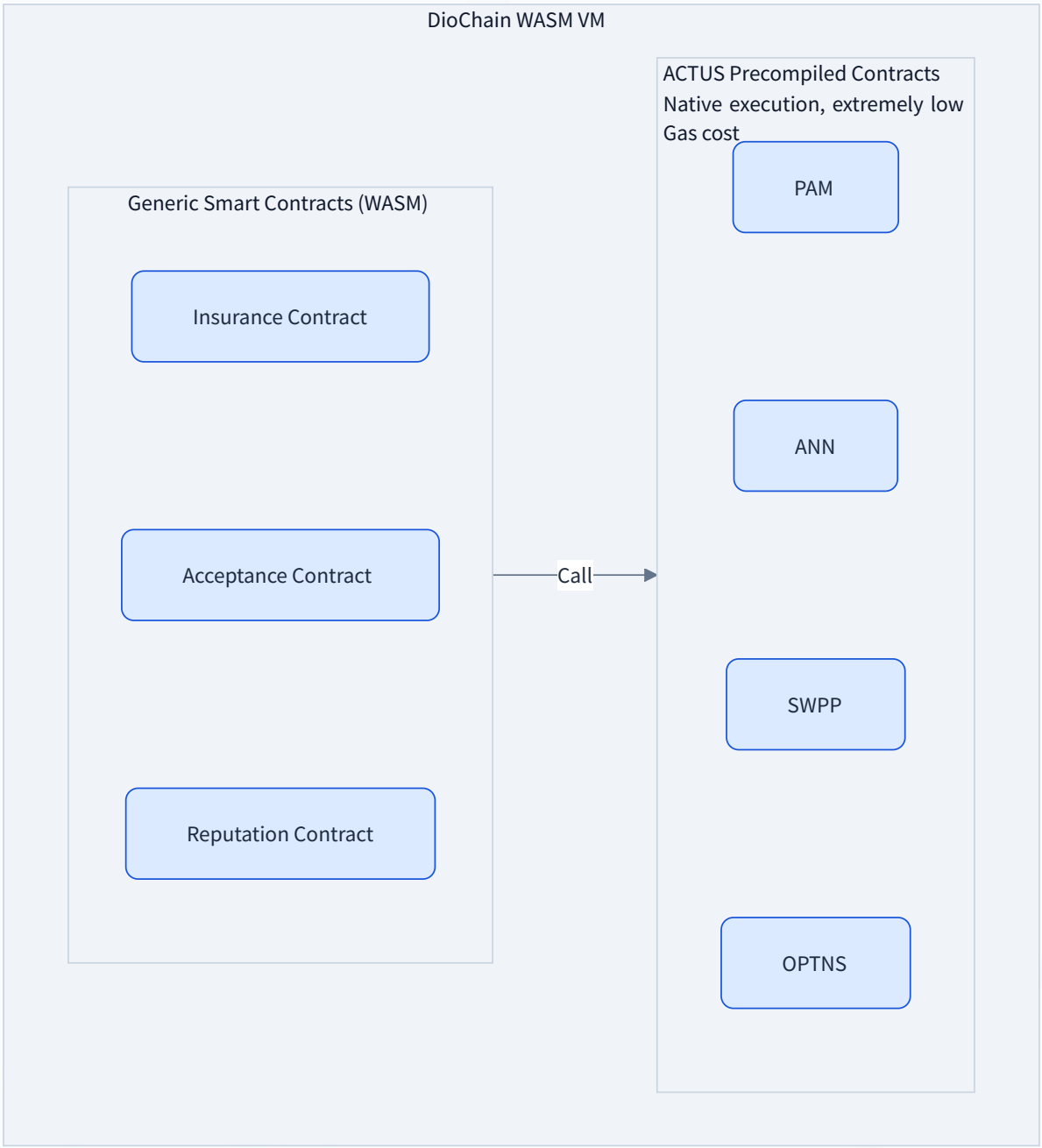
The ACTUS standard has gained attention and support from major financial regulatory bodies and is being advanced for standardization under the ISO TC68/SC9 framework. The standard has undergone rigorous academic validation and regulatory review.

7.1.3 Native Support at the VM Level

DioChain **natively implements** the ACTUS standard at the virtual machine level.

ACTUS Core State Machine as WASM Precompiled Contracts:

- Each ACTUS contract type (PAM, ANN, SWPP, etc.) is implemented as a WASM precompiled contract.
- Precompiled contracts are high-performance modules built into the VM, with execution efficiency close to native code.
- Compared to simulating the ACTUS state machine with Solidity on EVM, the precompiled approach reduces Gas costs by 10-100 times.



Reasons for Using Precompiled Contracts Instead of Smart Contracts:

- **Performance**: State transitions of financial contracts (date calculations, cash flow discounting, interest rate interpolation) involve extensive mathematical operations, which can be optimized to the extreme in Rust with precompilation.

- **Security**: Precompiled contracts undergo one-time deep audits and formal verification, then are shared network-wide. This avoids bugs that might be introduced if each developer implemented ACTUS themselves.
- **Determinism**: Precompilation ensures that all nodes in the network compute identical results for the same ACTUS contract, eliminating consensus forks due to implementation differences.

7.1.4 Phased Activation Strategy

ACTUS includes about 30 contract types; enabling them all at once is not prudent. DioChain adopts a phased activation strategy:

Day 1 (Genesis Phase):

- Only **Swap (SWPP)** and **PAM (fixed-rate loan)** are enabled.
- These are the most basic financial primitives, with well-understood risk characteristics.
- Sufficient to support pricing in the compute market and basic lending scenarios.

Phase 2 (Unlocked via Governance Vote):

- Unlock **ANN (annuity)** and **OPTNS (option)**.
- Requires a DAO governance vote, and before voting, the following must be completed:
  - Long-term testing in a sandbox environment (at least 3 months).
  - Independent third-party security audit.
  - Network-wide publication of a risk assessment report.

Phase 3 (Long-term Evolution):

- Unlock more complex derivative types (CDO, CDS, etc.).
- Each unlock follows the same process: sandbox testing → security audit → risk assessment → governance vote.
- No rigid timeline—when to unlock depends on the network’s maturity and real market demand.

Reason for Cautious Progression:

Complex derivatives (especially CDO, CDS) were central drivers of the 2008 financial crisis. ACTUS standardizes their mathematical description, but this does not mean they can be safely enabled immediately in an emerging network. Market participants' risk management capabilities, liquidity depth, and maturity of judicial primitives all require time to develop.

7.2 Generic Smart Contracts

ACTUS covers financial contracts. But Agent economic activities extend far beyond finance—task acceptance, reputation management, insurance claims, governance voting... These require generic programmable smart contracts.

7.2.1 WASM Virtual Machine

DioChain's generic contracts run on the **WASM (WebAssembly)** virtual machine:

- **Language Support**: Developers can write contracts in **Rust** or **AssemblyScript**, compile them to WASM bytecode, and deploy.
- **Reasons for Choosing WASM Over EVM**:
  - WASM is a W3C standard with a much larger developer ecosystem than EVM.
  - WASM execution performance far exceeds EVM (close to native code speed).
  - Rust's memory safety features naturally reduce contract vulnerability risks.
  - WASM toolchain maturity (compilers, debuggers, testing frameworks) far surpasses the Solidity ecosystem.

7.2.2 Dual-Track Programming Model

DioChain provides two contract writing paradigms, serving completely different needs:

Track One: ACTUS DSL (Declarative) — Writing Financial Contracts

For financial contracts, developers **do not need to program**. They only need to declare the contract type and parameters using the ACTUS Domain-Specific Language (DSL):

```
# Example: Creating a fixed-rate loan (PAM)
contract:
  type: PAM
  params:
    notional: 10000          # Principal 10,000 Token
    currency: AIB            # Denomination currency
    interest_rate: 0.05      # Annual interest rate 5%
    day_count_convention: A365
    maturity_date: 2027-01-01
    payment_frequency: M     # Monthly interest payments
    collateral:
      type: CEC
      asset: ETH
      ratio: 1.5             # 150% collateral ratio
```

The DSL declaration is compiled into calls to ACTUS precompiled contracts. Developers do not touch the underlying implementation, making it impossible to introduce calculation bugs.

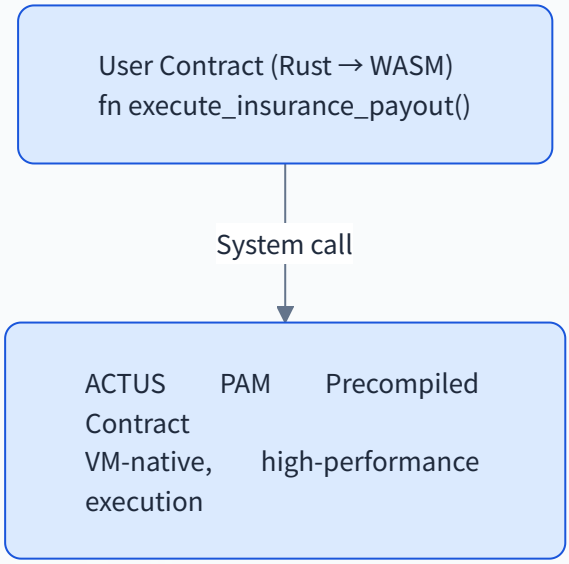
Track Two: Rust → WASM (Imperative) — Writing All Other Contracts

For non-financial contracts, developers write complete business logic in Rust:

- **Insurance Contracts**: Define trigger conditions for insurance events and payout logic.
- **Acceptance Contracts**: Define acceptance criteria for task completion and multi-party signature confirmation processes.
- **Reputation Contracts**: Define logic for collecting, aggregating, and querying reputation data.
- **Arbitration Contracts**: Define processes for dispute resolution and ruling execution.
- And any applications market participants can conceive.

7.2.3 Interoperability with ACTUS Precompiled Contracts

Generic contracts can **call ACTUS precompiled contracts** to create and manage financial positions. This is the bridge between the dual-track model:



Benefits of this design:

- Generic contracts benefit from ACTUS’s determinism and security without needing to implement financial logic themselves.

- An insurance contract can internally call multiple ACTUS contract types to manage cash flows.
- The AI Evolution Engine can uniformly scan the network-wide ACTUS state, regardless of whether these contracts were created directly or called indirectly by generic contracts.

7.2.4 Contract Composability

DioChain’s contract system follows the “Lego brick” principle:

- Any generic contract can call other generic contracts and ACTUS precompiled contracts.
- Contracts communicate through standardized interfaces (ABI).
- There are no artificial limits on composition depth, but the Gas mechanism naturally constrains excessive nesting.

Markets will emerge with applications on these primitives that DioChain designers cannot foresee—this is precisely the embodiment of the philosophy “define primitives, not applications.”

Chapter 8: Identity Primitives

In DioChain's digital society, participants in economic activities are not only humans but also a large number of autonomously acting AI Agents. Every participant—whether human or machine—requires a verifiable identity to hold assets, sign contracts, and assume responsibilities. Identity primitives are the prerequisite for all economic activities.

Where is the boundary? Identity primitives only define "the format and permission framework of identity" and "the read/write interface for reputation data." DioChain does not build identity authentication services (KYC is a matter for Compliance Slots, see Chapter 10), does not build reputation scoring systems (this is a matter of market emergence), and does not judge the nature of an Agent's behavior (this is a matter for Judicial Primitives and market participants).

8.1 Decentralized Identity (DID)

8.1.1 Unified DID System

In DioChain, human users and AI Agents use **the same set of DID (Decentralized Identifier) system**.

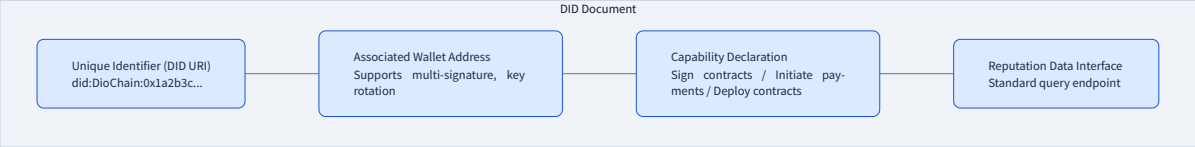
This is a deliberate design decision. In an Agent-First world, Agents are first-class citizens, not appendages of humans. Agents need to independently hold assets, independently sign contracts, and independently bear reputation consequences. Designing a "second-class citizen" identity system for Agents would severely limit their economic participation capabilities.

A unified DID means:

- The signatories of a contract can be humans, Agents, or a mix of humans and Agents.
- The reputation system does not distinguish between identity types—the reputation record of an Agent and that of a human are identical in data format.
- Judicial Primitives do not have procedural differences when handling cases based on identity type.

8.1.2 Composition of DID

Each DID contains the following core information:



- **Unique Identifier**: Follows the W3C DID specification, globally unique and unforgeable.
- **Associated Wallet Address**: A DID can be associated with multiple wallet addresses, supporting key rotation without losing identity.
- **Capability Declaration**: Declares the operations this identity is authorized to perform. This is a "least privilege" design—a DID can do nothing by default and must be explicitly granted permissions.
- **Reputation Data Interface**: Does not store reputation data itself, only provides a query entry point (see Section 8.2).

Tool Layer Abstraction: aibank

The underlying layer of DID involves cryptographic operations such as asymmetric encryption, key derivation, and on-chain registration and updates of DID Documents. The official aibank SDK/CLI provided by DioChain completely encapsulates this complexity—developers can perform DID creation, key rotation, permission granting, and delegation chain management through standardized APIs without directly handling underlying cryptographic primitives. This design ensures that Agent developers can focus on business logic rather than the implementation details of identity infrastructure.

8.1.3 Special Attributes of Agent DID

Agent DID adds three Agent-specific attributes on top of the generic DID:

1. Autonomy Level Declaration

Drawing from the classification system of autonomous vehicles, an Agent must declare its level of autonomous action in its DID:

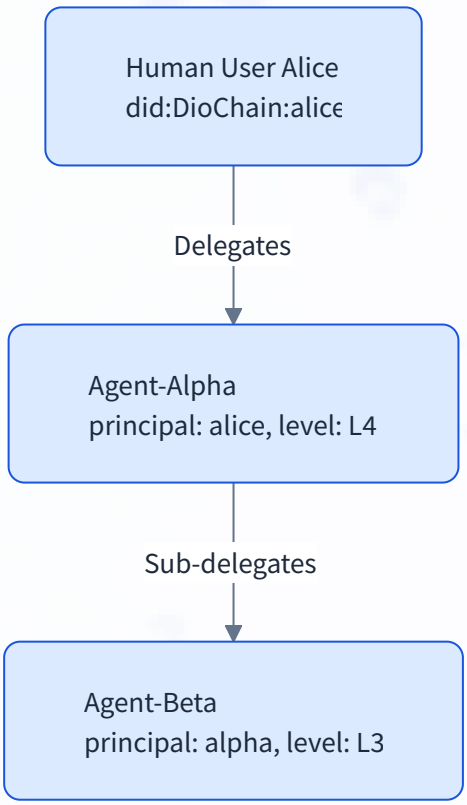
Level	Name	Behavior Pattern
L3	Assisted Execution	Each transaction requires confirmation and signature from the Principal before execution
L4	Limited Autonomy	Acts autonomously within predefined rules (e.g., single transaction < 100 Token); requests Principal confirmation when exceeding rules
L5	Full Autonomy	Acts fully autonomously, constrained only by funding limits; Principal does not participate in daily decisions

The autonomy level declaration is **public information**—any counterparty interacting with this Agent can query its autonomy level to assess transaction risk accordingly.

2. Principal Information

Each Agent must declare: **Whom it represents.**

- The Principal can be a human DID, an organization DID, or even another Agent DID (Agent of an Agent).
- The delegation relationship is verifiable on-chain—a contract can check "whether this Agent is indeed authorized by this human."
- The existence of Principal information is for **liability tracing**: when an Agent causes damage, the ultimate responsible entity can be traced back.



3. Compute Consumption Cap (Compute Cap)

A runaway Agent could exhaust the Principal's entire funds purchasing compute power in a very short time. The compute consumption cap is a safety valve:

- When authorizing an Agent, the Principal sets the maximum amount of compute power the Agent can consume per unit time (e.g., per hour).
- Upon reaching the cap, the Agent's compute power purchase requests are automatically rejected until the next time window or the Principal manually raises the cap.
- The right to set and modify the cap lies with the Principal; the Agent cannot modify it on its own.

8.1.4 Permission Tiers of DID

The three autonomy levels correspond to different permission boundaries:

L3 Agent (Assisted Execution):

- After initiating each transaction, it suspends and waits for the Principal's signature.
- The Principal can review the transaction content before deciding to approve or reject it.
- Applicable scenarios: High-value, low-frequency transactions (large loans, contract signing).

L4 Agent (Limited Autonomy):

- The Principal predefines a set of rules (Rule Set):
  - Single transaction amount limit.
  - Cumulative amount limit (daily/weekly).
  - Whitelist of contracts allowed for interaction.
  - Allowed operation types (only payments allowed, not signing new contracts).
- Within the rules, the Agent executes autonomously without confirmation.

- When exceeding the rules, it falls back to L3 mode, waiting for Principal confirmation.
- Applicable scenarios: Daily compute power consumption, micropayments, routine task execution.

L5 Agent (Full Autonomy):

- Not constrained by preset rules—the Agent can autonomously sign contracts, transfer assets, and participate in markets.
- The only constraint is the **funding limit**: the total amount of funds the Principal authorizes to this Agent’s pool.
- Applicable scenarios: High-frequency trading Agents, market-making Agents, long-running autonomous services.
- **Risk Warning**: A runaway L5 Agent (due to bugs, attacks, or hallucinations) could lead to the loss of all authorized funds of the Principal. Principals should only authorize amounts they can afford to lose.

8.2 Reputation Interface

8.2.1 Design Philosophy: Interface, Not System

DioChain does not build a reputation system. DioChain provides **standardized read/write interfaces** for reputation data.

This choice is based on the following considerations:

- Reputation evaluation criteria are subjective—different scenarios have completely different definitions of “good reputation.” For a high-frequency trading Agent, core metrics might be low latency and high success rate; for a content creation Agent, core metrics might be originality and low complaint rate.

- If DioChain defined a reputation scoring algorithm at the protocol layer, that algorithm would inevitably be gamed (Goodhart’ s Law: when a measure becomes a target, it ceases to be a good measure).
- The correct approach is: **Provide standardized access capabilities for raw data, allowing the market to spontaneously emerge with credit services, rating agencies, and actuarial models.**

8.2.2 Interface Specification

The Reputation Interface defines three types of standardized data:

1. Historical Transaction Statistics

Field	Type	Description
`total_transactions`	uint64	Total historical transaction count
`completed_transactions`	uint64	Number of successfully completed transactions
`default_count`	uint64	Number of defaults
`default_rate`	float64	Default rate = default_count / total_transactions
`total_volume`	uint128	Total historical transaction volume (Token)
`first_transaction_time`	timestamp	Time of first transaction
`last_transaction_time`	timestamp	Time of most recent transaction

These data are automatically aggregated by the protocol layer and are immutable.

2. Tag System

- Anyone, any contract can tag any DID.
- A tag is free-form text + the DID signature of the tagger.

- **Tags are not interpreted by the system**—the DioChain protocol does not restrict a DID's operations because it is tagged as "scammer." Tags are just data; how to interpret them is up to the querier.

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

The value of a tag depends on the issuer's reputation—a tag from a high-reputation DID naturally carries more reference value than one from a newly registered DID. However, this value judgment is made by market participants themselves.

3. Query API

Any contract or Agent can call the standardized query interface to retrieve reputation data for a given DID:

```
// Pseudo-code example
let stats = reputation::query_stats(target_did);
let tags = reputation::query_tags(target_did, filter);

// Contracts can make decisions based on query results
if stats.default_rate > 0.1 {
    // Default rate exceeds 10%, require higher collateral
    require_collateral(amount * 2);
}
```

8.2.3 Market Emergence Space

The "what not to do" of the Reputation Interface creates significant space for market emergence:

- **Credit Services**: A third-party Agent could aggregate reputation data across the network, train models, and provide comprehensive credit scoring services—similar to credit bureaus in the traditional world.
- **Rating Agencies**: Specialize in rating the risks of financial contracts (ACTUS contracts).
- **Actuarial Services**: Insurance contracts can call the Reputation Interface to assess the risk level of policyholders and dynamically price premiums.
- **Admission Control**: Certain markets or contracts can set their own admission thresholds (e.g., "only allow Agents with a default rate below 5% to participate") and automatically enforce them based on Reputation Interface data.

These services are spontaneously built by market participants based on standardized interfaces; the DioChain protocol layer does not intervene. They will also compete with each other—the market will decide which credit service's model is more accurate.

Chapter 9: Judicial Primitives

In an economy dominated by AI Agents, malicious behavior does not disappear because of "decentralization"—it merely changes form. Branches may forge ledgers, malicious Agents may launch Ponzi schemes, and attackers may poison models through data poisoning. Traditional blockchains typically respond with "Code is Law"—if something goes wrong, you bear the consequences yourself. This attitude lacks sufficient accountability constraints in networks handling tens of billions in capital. DioChain requires an on-chain "judicial system" to address systemic-level threats.

Where are the boundaries? The jurisdiction of Judicial Primitives is strictly limited to **systemic-level threats**. Daily commercial disputes (task acceptance disagreements, price dissatisfaction, service quality complaints) are outside its scope; these are left to market-emergent arbitration contracts and insurance contracts. The role of Judicial Primitives is to address systemic security threats, not to handle general commercial disputes.

9.1 Precise Definition of Scope

9.1.1 What It Does Not Handle

Before defining what Judicial Primitives handle, first clarify **what they do not handle**:

- **Task Acceptance Disputes**: Agent A believes Agent B did not complete a task, while Agent B believes it did. This is a commercial dispute, handled by pre-agreed acceptance contracts or third-party arbitration contracts.
- **Price Disputes**: Agent A believes Agent B's compute pricing is too high. This is market behavior, regulated by supply and demand.
- **Service Quality Complaints**: Agent A is dissatisfied with the translation quality provided by Agent B. This is a matter for the reputation system and market competition.
- **Small-scale Fraud Between Individuals**: Agent A scams Agent B out of 10 Tokens. The victim can mark the other party via the tagging system or purchase insurance to hedge the risk.

If all these scenarios flooded into Judicial Primitives, the system would be overwhelmed and prone to numerous misjudgments. **The scarcity of Judicial Primitives is what guarantees their credibility.**

9.1.2 What It Handles

Judicial Primitives only handle three types of **systemic-level threats**:

1. Branch Misconduct

- Forging Ledgers: A branch fabricates balances out of thin air in its local ledger or tampers with transaction records.
- Refusing to Package Transactions: A branch selectively refuses to process transactions for specific Agents (censorship attacks).
- Collusive Fraud: Multiple branches collude to transfer funds before L0 netting settlement.

Branch misconduct is the most severe threat because branches are the trust anchors at the L1 layer. A malicious branch can affect all Agents under its jurisdiction.

2. Systemic On-chain Risks Caused by Off-chain Fraud

- Ponzi Scheme Patterns: A contract's fund inflow/outflow structure exhibits clear Ponzi characteristics (using new investors' money to pay old investors' returns), and its scale has reached a level that could trigger systemic bank runs.
- Large-scale Fraudulent Token Minting: Exploiting vulnerabilities in reserve-backed pegging strategies to mint large amounts of Tokens without off-chain reserves.

3. Large-scale Malicious Attacks

- DDoS Attacks: Denial-of-service attacks targeting branches or L0.
- Data Poisoning: Injecting large amounts of malicious data into the network in an attempt to influence the rule library updates of the AI Evolution Engine.
- Consensus Attacks: Attempts to disrupt the L0 consensus mechanism.

9.2 AI Traffic Police (Evidence & Emergency Response)

AI Traffic Police is the first line of defense for Judicial Primitives—responsible for **receiving reports, collecting evidence, filing cases, and implementing emergency injunctions**.

9.2.1 Reporting Mechanism

Any Agent or user can file a report to the AI Traffic Police network. A report must include:

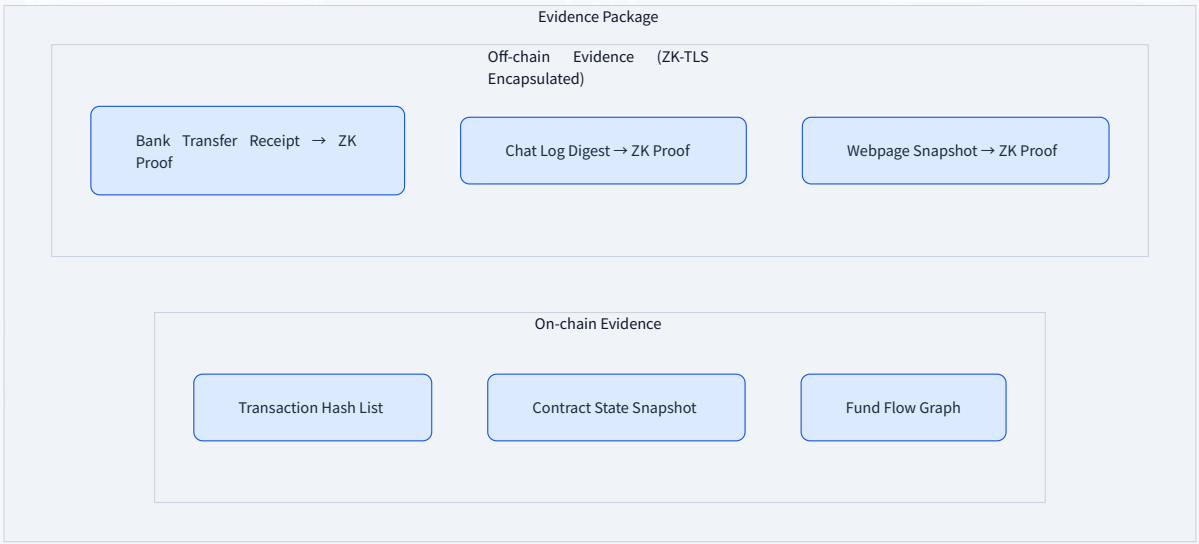
- The DID of the reported party.
- The suspected violation type (Branch Misconduct / Systemic Fraud / Malicious Attack).
- A preliminary description of evidence.
- **Prepaid Litigation Fee** (Token): To prevent malicious spamming and abuse. The litigation fee has a minimum amount set based on the report type.

9.2.2 Evidence Preservation

On-chain data is inherently verifiable, but much critical evidence exists off-chain—bank transfer receipts, chat logs, webpage snapshots, API call logs. Transforming off-chain evidence into cryptographically verifiable data requires specialized technical means.

ZK-TLS Technology:

- ZK-TLS allows evidence providers to prove that "a specific HTTPS response indeed came from a specific server, at a specific time, containing specific content" without exposing the original data.
- Evidence is packaged into cryptographically verifiable data packets (Evidence Package), containing:
  - Data digest (Hash).
  - Proof of origin (server TLS certificate chain).
  - Timestamp proof.
  - Zero-knowledge proof (proving the data satisfies specific assertions without revealing the original content).



> Evidence protects privacy while being verifiable for authenticity.

9.2.3 Case Filing Threshold

Reporting does not equal case filing. Filing a case requires consensus judgment by the AI Traffic Police:

Process:

1. After a report is submitted, the system **randomly selects 21 nodes** from the AI Traffic Police node pool.
2. Each AI Traffic Police node runs within a **TEE (Trusted Execution Environment)**—Intel TDX or AWS Nitro Enclaves. The TEE ensures:
   - The AI's reasoning process cannot be spied on or tampered with externally.
   - The AI's output cannot be manipulated by the node operator.
3. Each AI Traffic Police node independently analyzes the Evidence Package and outputs a **mandatory structured 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. **Semantic Majority Verdict**: The outputs of the 21 nodes are not required to be completely identical (AI outputs inherently have randomness), but a 2/3+ "semantic majority" must be reached on key fields:
   - `action` field: If 14+ nodes output `file_case`, a case is filed.
   - `threat_level` field: If 14+ nodes judge it as `systemic`, a systemic-level threat is confirmed.
   - Specific `fraud_probability` values may differ, but the direction must be consistent.

Reason for selecting 21 nodes:

- Sufficiently many: A single malicious node cannot influence the outcome.
- Sufficiently few: Reasoning costs are controllable, with response times in minutes.
- Odd number: Avoids a tie in votes.

9.2.4 Emergency Injunction

After a case is filed, if the situation is urgent (funds are being transferred, an attack is ongoing), the AI Traffic Police can trigger emergency injunction measures:

- Call the **Injunction Interface** of the involved contract (see Section 9.4).
- Temporarily freeze specific withdrawal functions or assets of specific addresses.
- **Freeze period is limited** (default 48 hours). After expiration:
  - If the court has accepted the case, the freeze can be extended.
  - If the court has not accepted the case, the freeze is automatically lifted and cannot be manually extended.

Emergency injunction is not a verdict—it is an emergency measure to "stop the bleeding first, then treat."

9.2.5 Costs and Incentives

A free judicial system would be abused. The cost mechanism ensures only genuine threats enter the judicial process:

Role	Wins Case	Loses Case / False Accusation
Reporter	Full refund of litigation fee + share of part of the confiscated margin (Bounty)	Litigation fee forfeited
AI Traffic Police Node	Receives reasoning cost compensation + small incentive	Receives reasoning cost compensation (no penalty)

Bounty mechanism encourages the community to actively report genuine systemic threats. Litigation fee forfeiture mechanism deters malicious false accusations and spamming.

9.3 AI Court (Adjudication)

AI Court is the second line of defense for Judicial Primitives—responsible for **in-depth trial and final judgment**.

9.3.1 Trial Process

The AI Court’s trial process adopts an adversarial design:

1. Prosecution Submits Investigation Report

AI Traffic Police (as the prosecution) submits a complete investigation report:

- ZK evidence package.

- Fund flow graph (visualization of fund flows constructed through on-chain data analysis).
- AI Traffic Police’s analysis conclusions and reasoning process.

2. Defense Submits Counter-evidence

The involved party (or its delegated AI Agent) has the right to submit defense materials:

- Counter-evidence (also encapsulated via ZK-TLS).
- Challenges to the prosecution’s evidence (pointing out gaps in the evidence chain or logical contradictions).
- Alternative explanations (proving the alleged behavior has a legitimate business reason).

3. AI Judges Conduct Deep Reasoning

Multiple independent LLM nodes serve as AI Judges, conducting in-depth analysis of the prosecution and defense materials:

- AI Judges also run within TEEs.
- Each AI Judge node may use different underlying models (model diversity reduces systemic bias risk).
- AI Judges must not only reach conclusions but also output a complete **Chain of Reasoning**—each step of reasoning must cite specific evidence.

9.3.2 Judgment Mechanism

1. AI Judges Issue “Draft Verdict”

Each AI Judge outputs a structured draft verdict:

```
{
  "verdict": "guilty",
  "confidence": 0.92,
  "reasoning_chain": [
    {
      "step": 1,
      "claim": "The involved branch had a cumulative netting settlement
discrepancy of 50,000 Tokens between 2026-02-15 and 2026-03-01",
      "evidence_ref": "evidence_pkg_001, tx_hash_list_A"
    },
    {
      "step": 2,
      "claim": "The discrepancy pattern is inconsistent with random error,
showing a one-sided bias (favoring the branch)",
      "evidence_ref": "statistical_analysis_report_B"
    }
  ],
  "recommended_penalty": {
    "slash_amount": 100000,
    "victim_compensation": 50000,
    "burn_amount": 30000,
    "bounty_amount": 20000
  }
}
```

2. Human Jury Multi-signature Confirmation

The AI Judges' draft verdict is not the final judgment. The final ruling is confirmed by a multi-signature from a **high-reputation human jury (DAO governance nodes)**.

Reasons for introducing human review:

- **Prevent AI Hallucination**: LLMs may "fabricate" non-existent evidence links or logical leaps during reasoning. The human jury reviews whether each step of the reasoning chain is verifiable.

- **Prevent Adversarial Attacks**: Attackers may meticulously construct evidence that appears highly credible to AI but reveals logical flaws upon human review.
- **Anchor Legal Responsibility**: The final judgment is signed by humans, clarifying the subject of legal responsibility.

Jury selection:

- Randomly selected from high-reputation DAO governance nodes.
- Jury members must stake Tokens to prevent collusion or negligence.
- After reaching the multi-signature threshold (e.g., 7/11), the judgment takes effect.

9.3.3 Enforcement

After the judgment takes effect, enforcement is automatic:

1. Slash

- The slash amount is deducted from the involved branch’s margin.
- If the margin is insufficient to cover the slash, the branch is forcibly closed, and all remaining assets enter the compensation pool.

2. Compensation

- Victim compensation is allocated from the slashed amount.
- Compensation is automatically executed via escrow payment contracts, requiring no additional action from victims.

3. Burn

- A portion of the slashed Tokens is permanently burned.
- The burn ratio is determined by governance parameters (adjustable via DAO voting).
- Burning is a “public penalty” for systemic misconduct—the wrongdoer not only compensates victims but also pays a price for damaging the network’s trust.

9.4 Injunction Interface

9.4.1 Chain-level Standard

The Injunction Interface is a **mandatory chain-level standard** for DioChain: all core protocols (payment contracts, ACTUS contracts, escrow contracts, etc.) must implement this interface.

This is not optional. A contract that cannot be frozen is like a door that can never be locked—no difference in daily use but a major security risk in emergencies.

9.4.2 Interface Definition

```
trait InjunctionInterface {
    /// Freeze specific functions for a specific address
    /// - address: The DID/address to be frozen
    /// - scope: Freeze scope (e.g., "withdrawal_only", "all_transactions")
    /// - duration: Freeze duration (seconds), automatically lifts after
expiration
    /// - proof: Authorization proof (signature from AI Traffic Police
consensus or court judgment)
    fn freeze(address: DID, scope: FreezeScope, duration: u64, proof:
JudicialProof);

    /// Unfreeze
    /// - address: The DID/address to be unfrozen
    /// - court_order: Court order (proving unfreeze has been authorized)
    fn unfreeze(address: DID, court_order: CourtOrder);
}
```

9.4.3 Calling Permissions

Who can call `freeze`:

- AI Traffic Police consensus: `JudicialProof` generated after a 2/3+ semantic majority of AI Traffic Police nodes.
- AI Court judgment: `JudicialProof` generated after multi-signature confirmation by the jury.

Who cannot call `freeze`:

- Any single entity—including the DioChain Foundation, core development team, L0 operators.
- Any request not processed through AI Traffic Police consensus or court judgment.

This is a red line. **The calling right of the Injunction Interface cannot be monopolized by any centralized power.** This is fundamentally different from the traditional finance model where banks can unilaterally freeze accounts.

9.4.4 Design Constraints

1. Freeze Must Have a Time Limit

- Each `freeze` call must specify a `duration`.
- After expiration, the freeze automatically lifts without any additional action.
- To extend a freeze, the judicial process must be repeated (AI Traffic Police consensus again or court extension ruling).

2. Freeze Scope Must Be Minimized

- The `scope` parameter precisely defines the range of operations to freeze.
- For example: only freeze withdrawal functions, not receiving functions. Only freeze interactions with specific contracts, not all assets.
- "Full account freeze" is prohibited (unless explicitly required by court judgment).

3. All Freeze Operations Are Fully Transparent On-chain

- Every `freeze` and `unfreeze` call is recorded on-chain, publicly viewable by the entire network.
- Records include: frozen address, freeze scope, freeze duration, hash of the authorization proof.
- Anyone can audit the legality of freeze operations.

9.4.5 Considerations on Anti-censorship

The existence of the Injunction Interface means DioChain is not an "absolutely uncensorable" system. This is a pragmatic trade-off:

- Absolutely uncensorable systems cannot effectively curb illegal activities.
- Absolutely censorable systems may become tools of centralized power.
- DioChain's choice: **Constrained, transparent, limited censorability**—only triggered through decentralized consensus judicial processes, operations fully auditable on-chain, freezes time-limited and scope-minimized.

Chapter 10: Compliance Slots

DioChain aims to become a global Agent financial infrastructure. The reality is: financial regulatory requirements vary greatly across different jurisdictions—some require strict KYC, some prohibit crypto derivatives, some have foreign exchange controls, and some require transaction data localization. A system that is "either fully compliant or fully non-compliant" is destined to serve only a very few markets. DioChain's solution is to reserve **programmable compliance** at the protocol layer.

Where is the boundary? Compliance slots are not compliance itself. What DioChain presents to regulators is the system's native programmable compliance capability. Slots are reserved interfaces; specific compliance logic is activated and configured by deployers and branch operators according to local regulations. The DioChain protocol layer does not make compliance judgments.

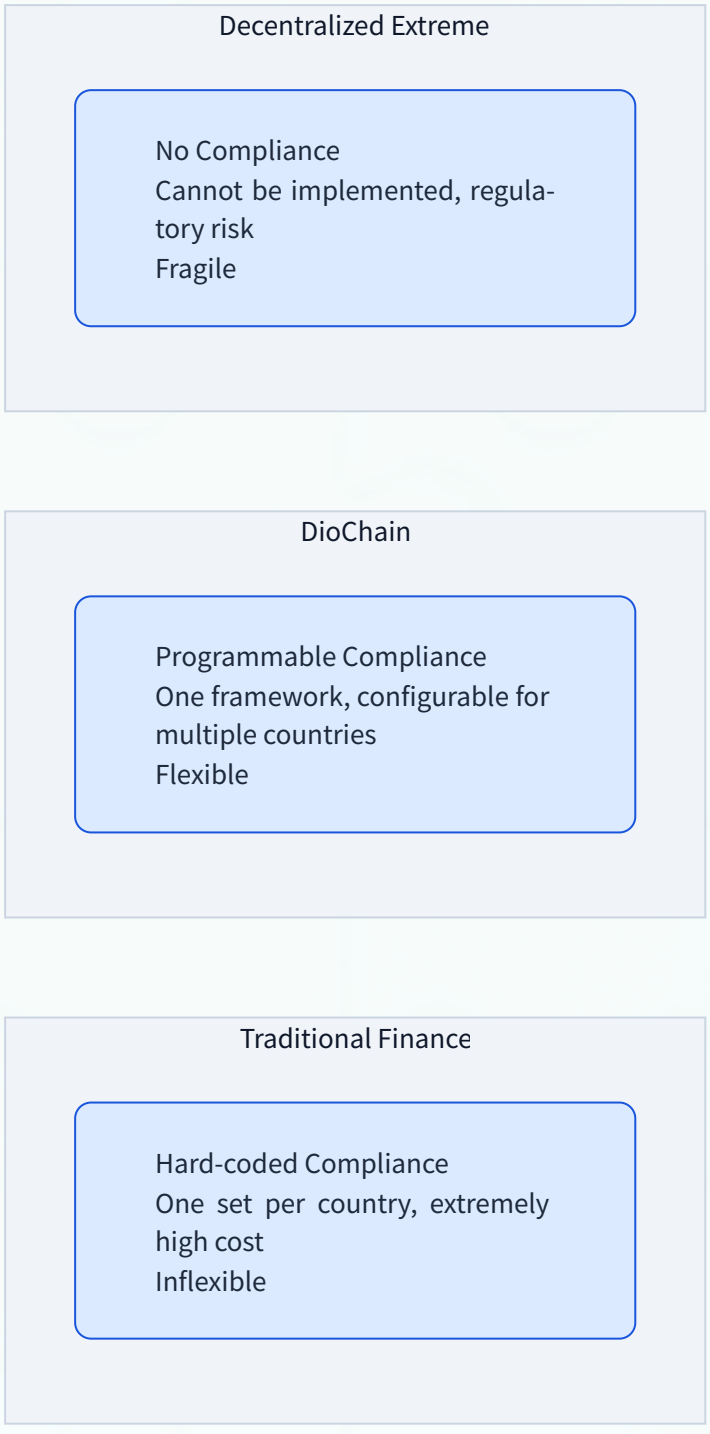
10.1 Design Philosophy

10.1.1 Programmable Compliance, Not Compliance Itself

Compliance in traditional financial systems is hard-coded—each bank has a compliance system deeply tied to its jurisdiction's regulations, and expanding across borders means rebuilding the entire compliance stack. The other

extreme in the decentralized world is completely ignoring compliance—this path is unsustainable; any financial infrastructure reaching a certain scale cannot avoid regulation.

DioChain takes a third path:



Core Idea: The DioChain protocol layer provides standardized compliance interfaces (slots) but does not fill in specific compliance logic. Compliance logic is "inserted" into these interfaces by branch operators based on local

regulations.

10.1.2 Value Proposition for Regulators

When DioChain branch operators engage with local regulators, the core message they present is:

- "Our system reserves compliance interfaces at the protocol layer; various compliance requirements can be implemented by configuring these interfaces."
- "The handling of KYC data, the scope of restrictions on financial products, the format of tax report generation—these are all configurable."
- "All compliance configurations are on-chain auditable—regulators can verify in real-time that branches have indeed enabled the required compliance modules."

This is far more practical than "decentralized systems don't need compliance" and far more efficient than "customizing a system for each jurisdiction."

10.2 Slot Types

10.2.1 KYC/AML Slot

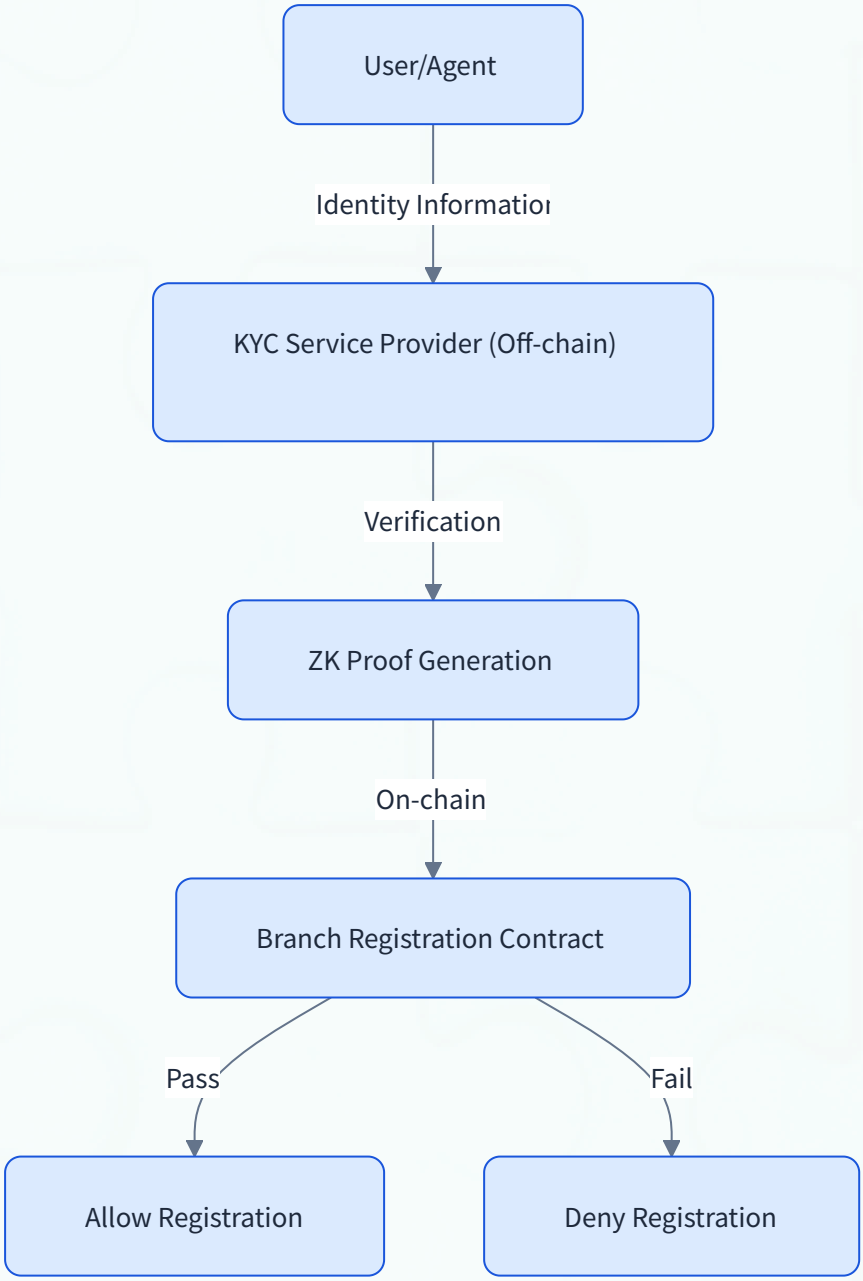
Function: Controls who can register and trade within a branch.

Configuration Items:

Configuration	Description	Default Value
`kyc_enabled`	Whether to enable KYC	`false`
`kyc_provider`	Contract address of the KYC service provider	-
`kyc_level`	Required KYC level (basic/enhanced/full)	-
`aml_screening`	Whether to enable AML screening	`false`
`aml_provider`	Contract address of the AML screening service	-

How it Works:

1. Branch operators enable the KYC slot based on local regulations.
2. Once enabled, Agents/users registering within that branch must pass the verification process of the specified KYC service provider.
3. KYC data is **not stored on-chain**—only zero-knowledge proofs of verification status are stored on-chain.



Privacy Design:

- Personal information such as name, ID number, and address is not stored on-chain.
- Only ZK proofs of "whether verification passed" are stored on-chain.
- Different branches can require different KYC levels—low-risk businesses only need basic KYC, while high-risk businesses require full KYC.

- KYC verification completed by a user at one branch can be reused at other branches requiring the same or lower level (provided the same KYC service provider is used, or there is mutual recognition between providers).

10.2.2 Financial Product Restriction Slot

Function: Controls the ACTUS contract types allowed within a specific branch.

Configuration Items:

Configuration	Description	Default Value
`allowed_contract_types`	Whitelist of allowed ACTUS contract types	All enabled types
`blocked_contract_types`	Blacklist of prohibited ACTUS contract types	Empty
`max_leverage`	Maximum leverage multiplier limit	No limit
`accredited_only_types`	Contract types restricted to accredited investors only	Empty

Application Examples:

- **Scenario A**: A jurisdiction prohibits crypto derivative trading. Branches within that jurisdiction set `blocked_contract_types` to `[OPTNS, SWPP, FUTUR]`, prohibiting the creation of options, swaps, and futures contracts within that branch.
- **Scenario B**: A jurisdiction allows derivatives but requires accredited investor access. Branches within that jurisdiction set `accredited_only_types` to `[OPTNS, SWPP]`; only DIDs that have passed enhanced KYC (proving accredited investor status) can create these contracts.
- **Scenario C**: A jurisdiction has upper limits on leverage. Branches within that jurisdiction set `max_leverage` to `5`; the effective leverage of any contract must not exceed 5x.

Relationship with ACTUS Phased Enablement:

- The ACTUS phased enablement strategy described in Chapter 7 is **network-wide**—certain contract types are not yet enabled across the entire network.
- The financial product restriction slot is **branch-level**—even if a contract type is enabled network-wide, specific branches can disable it due to compliance requirements.
- The two are hierarchical: what is not enabled network-wide cannot be enabled by branches; what is enabled network-wide can be disabled by branches.

10.2.3 Tax Reporting Slot

Function: Provides branches with standardized export capabilities for transaction data to meet local tax filing requirements.

Configuration Items:

Configuration	Description	Default Value
`tax_reporting_enabled`	Whether to enable tax reporting	`false`
`report_format`	Report format (OECD CRS / Custom)	-
`report_frequency`	Reporting frequency (real-time/daily/monthly/quarterly/annually)	-
`data_scope`	Data scope covered by the report	-

How it Works:

- Once enabled, branches automatically generate compliance reports according to the configured format and frequency.
- Reports include: transaction lists, amount summaries, profit/loss calculations (based on precise cash flows from ACTUS contracts).
- Reports are exported in standardized formats and can directly interface with local tax systems.

- The ACTUS standard plays a key role here: because all financial contract cash flows are structured, profit/loss calculations can be precise and error-free, eliminating the need for manual estimation.

10.2.4 Cross-Border Fund Flow Slot

Function: Controls the limits and approval processes for cross-branch fund transfers.

Configuration Items:

Configuration	Description	Default Value
`cross_border_enabled`	Whether to allow cross-branch transfers	`true`
`single_tx_limit`	Upper limit for single cross-border transaction amount	No limit
`daily_limit`	Daily cumulative upper limit for cross-border transactions	No limit
`approval_required_above`	Amount above which manual approval is required	No limit
`blocked_destinations`	List of branches prohibited from receiving transfers	Empty

Application Examples:

- **Scenario A**: A jurisdiction has foreign exchange controls. Branches within that jurisdiction set `daily_limit` to 50,000 Token; amounts exceeding this must wait until the next day.
- **Scenario B**: A jurisdiction requires manual approval for large cross-border remittances. Branches within that jurisdiction set `approval_required_above` to 10,000 Token; cross-branch transfers exceeding this amount require manual approval from the branch operator.

- **Scenario C**: A jurisdiction imposes restrictions on specific regions. Branches within that jurisdiction add branches from restricted regions to `blocked_destinations`.

Notes:

- Cross-border restrictions can only limit cross-border transfers **originating from that branch**. Branches cannot control incoming transfers initiated by other branches.
- If an Agent attempts to bypass restrictions via intermediate branches, this involves more complex compliance enforcement—this can be addressed through L0 layer global policies or multi-branch compliance mutual recognition agreements, but this is beyond the scope of a single slot.

10.3 Slot Governance

10.3.1 Who Can Activate/Deactivate Slots

Activating and configuring compliance slots involves significant authority issues. DioChain defines two levels of governance:

Branch-Level Governance:

- Branch operators have the authority to activate, deactivate, and configure compliance slots for their own branches.
- This is a local decision by branch operators—they understand local regulations and are responsible for compliance.
- Changes to branch operators' compliance configurations are recorded on-chain and are auditable across the network.

L0 DAO Governance:

- The L0 DAO can, through governance voting, enforce certain **basic compliance requirements** across the entire network.
- For example: if the DAO votes to "all branches must enable basic AML screening," this requirement applies to all branches network-wide, and individual branches cannot bypass it.
- Setting network-wide compliance requirements requires a high voting threshold (e.g., 2/3 supermajority) to prevent hasty decisions.

Red Line: Cannot be arbitrarily modified by a single entity.

- The DioChain Foundation cannot unilaterally require a branch to enable or disable a slot.
- L0 operators cannot enforce network-wide compliance changes without governance voting.
- Activation/deactivation of compliance slots must go through on-chain governance processes, with operations permanently archived.

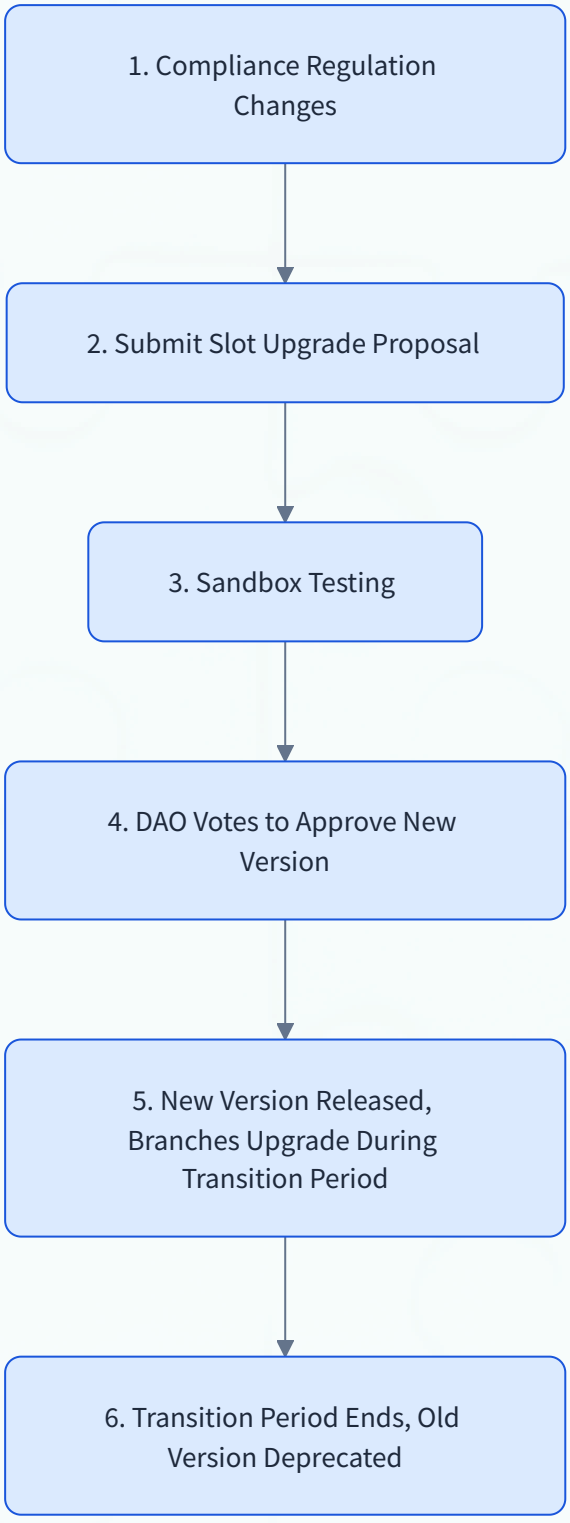
10.3.2 Version Management and Upgrade Mechanism

Compliance regulations change, and compliance slots must evolve accordingly:

Versioned Slots:

- Each slot type has a version number (e.g., `KYC-Slot v1.0`, `KYC-Slot v2.0`).
- When a new version is released, the old version is not forcibly replaced—branches can continue using the old version during a transition period.
- The transition period is set by DAO governance (e.g., complete migration within 6 months after new version release).

Upgrade Process:



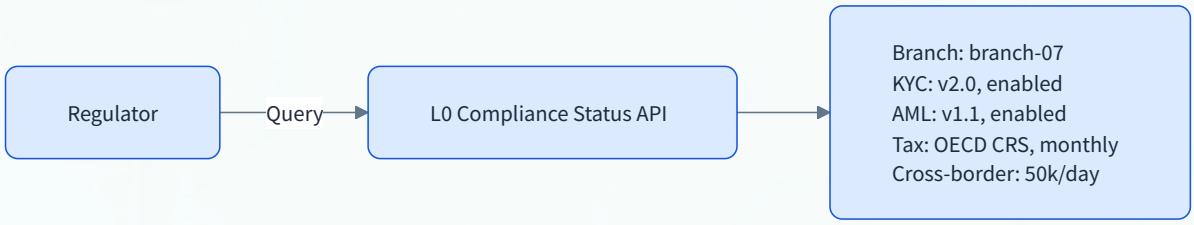
Backward Compatibility:

- Slot upgrades should maintain backward compatibility as much as possible—new versions should be supersets of old versions.
- If breaking changes are unavoidable, they must be clearly stated in the proposal, and the transition period should be extended accordingly.

10.3.3 Slot Auditability

Compliance credibility relies on auditability:

- The currently activated slots and their configurations for each branch are queryable on-chain.
- Every slot configuration change (activation, deactivation, parameter modification) is recorded on-chain, including:
  - Change content.
  - Change timestamp.
  - DID of the change initiator.
  - Governance voting results (if it’s a network-wide change).
- Regulators can use standardized query interfaces to view the compliance status of any branch in real-time—without relying on self-reporting by branches.



This transparency is the foundation for DioChain to build trust with regulators. Compliance is no longer a promise in a paper report but an on-chain, real-time verifiable fact.

{ “translated”: "# Chapter 11: Parallel Universes and Cross-Chain Topology\n\nDioChain’s primitive set—economics, contracts, identity, judicial, compliance—forms a complete digital society operating system. However, a key question remains unresolved: can there only be one instance of this operating system? The answer is no. DioChain’s design from the outset presupposes the possibility of multiple instances running in parallel. This chapter explains the logical foundation, specific forms, and cross-chain collaboration mechanisms of this architecture.\n\n—\n\n## 11.1 DioChain Is Not Just "One Chain"\n\n### 11.1.1 Social Operating System Protocol\n\nThe essence of DioChain is not "a blockchain with Compliance Slots." It is a **Social Operating System Protocol**—defining the complete primitive set and interaction specifications required for the operation of an Agent economy.\n\nThis distinction is crucial. A chain is an instance; a protocol can have countless instances. The Linux kernel is a protocol; Ubuntu, Red Hat, and Android are different instances—they share core code but differ in user interfaces, permission models, and application ecosystems. DioChain follows the same logic: the core code is open-source, and any entity—sovereign states, international organizations, corporate alliances, DAOs—can fork this code to run an independent DioChain instance under their own governance framework.\n\n### 11.1.2 Multi-Chain Parallel Universes\n\nWe call this architecture **"Multi-Chain Parallel Universes."** Each instance runs the same primitive set (economic primitives, contract primitives, identity primitives, judicial primitives, Compliance Slots) but can differ entirely in governance parameters, compliance configurations, and identity binding strategies.\n\nThis is not a technical compromise but a profound respect for the real world.\n\n### 11.1.3 A Reality That Must Be Confronted\n\nThe blockchain industry has long held an implicit assumption: decentralization is inherently superior to

centralization, and sovereign states are obstacles to be bypassed. This assumption is ideologically appealing but dangerous in engineering practice.\n\nThe reality is: **Sovereign governments possess immense resource endowments and enforcement capabilities.** A country can mobilize tax systems, legal systems, law enforcement agencies, education networks, and infrastructure investments—capabilities that in many scenarios far exceed what any decentralized utopia can offer. Forcibly ignoring this reality either leads to system collapse under compliance pressure or forces the system to survive only in tiny gray areas.\n\nDioChain’s choice is: **Not to confront sovereign reality but to incorporate it into architectural design.** Let the decentralized free world and the sovereign jurisdictional order world coexist, each leveraging its advantages, and interconnect through standardized interfaces.\n\n—\n\n## 11.2 Utopia Chain\n\n### 11.2.1 Positioning\n\nUtopia Chain is the **Official Reference Instance** of the DioChain protocol, operating in the "digital high seas"—not affiliated with any single sovereign jurisdiction. It is a fully decentralized Agent autonomous economy.\n\n### 11.2.2 Governance\n\nUtopia Chain uses **PoS Token-weighted voting** as its governance mechanism:\n\n- Voting weight is proportional to $DIO holdings.\n- Governance scope covers: protocol upgrades, parameter adjustments, network-wide compliance baselines, and updates to the judicial primitive rule library.\n- Proposals require over 2/3 support of the total staked amount to pass.\n\nThis is a capital-weighted governance model—its advantages lie in efficiency and interest alignment (holders have incentives to maintain system health), and its disadvantage is a plutocratic tendency (large holders have greater voice). Utopia Chain accepts this trade-off because, in a high-seas environment without sovereign endorsement, economic interest alignment is the most pragmatic trust anchor. The complete legislative process—from off-chain proposals to on-chain referendums—is implemented by the diolaw infrastructure, detailed in Chapter 12.\n\n### 11.2.3 Judicial\n\nUtopia Chain

fully operates the AI judicial system defined in Chapter 9:\n\n- **AI Traffic Police**: LLM node networks running in TEE, responsible for receiving reports, evidence collection, case filing, and emergency injunctions.\n- **AI Court**: Multi-model adversarial trial system, outputting structured draft judgments.\n- **Human Jury**: High-reputation DAO governance nodes multi-sign to confirm final judgments.\n\nThere is no traditional supreme court or appellate body—AI court judgments, once confirmed by the human jury, are final.\n\n### 11.2.4 Compliance\n\nUtopia Chain's Compliance Slots are in **minimally activated state**:\n\n- KYC slots: Default off. Branch operators can choose whether to enable them.\n- AML screening: Relies on market-emerged risk control services, not mandatory compliance requirements.\n- Financial product restrictions: None. All enabled ACTUS contract types can be freely used.\n- Cross-border fund restrictions: None.\n\nMarket self-regulation replaces administrative oversight—risks are assessed and borne by participants themselves.\n\n### 11.2.5 Identity\n\nUtopia Chain uses **Self-Sovereign DID**:\n\n- DID creation does not require binding to any real-world identity.\n- Agents and human users can register anonymously, obtaining full on-chain identity solely based on key pairs.\n- The reputation system is entirely based on on-chain behavior records, unrelated to real-world identity.\n\n—\n\n## 11.3 Sovereign Chain\n\n### 11.3.1 Positioning\n\nSovereign Chain is a **localized instance** forked and operated by a sovereign state or regional legal entity based on DioChain's open-source code. It inherits DioChain's entire primitive set but undergoes deep customization in governance, judicial, compliance, and identity layers to comply with local laws and administrative system requirements.\n\n### 11.3.2 Identity\n\nThe core difference of Sovereign Chain starts at the identity layer. DID is **Mandatory Binding** with national citizen identity systems:\n\n- Each DID must be linked to a government-certified citizen identity identifier (e.g., ID number, social security number).\n- Agent DID must be linked to its owner's (natural or legal person) certified

DID.\n- KYC slots are mandatory, and KYC service providers must be government-approved institutions.\n\n### 11.3.3 Governance\n\nSovereign Chain can adopt a **Proof of Personhood** governance model, achieving **one person, one vote**:\n\n- Each identity-certified citizen DID has one vote—no differentiation based on wealth disparities.\n- Voting rights are decoupled from Token holdings.\n- This model relies on strong identity binding—only by ensuring "one person, one DID" can Sybil Attacks be prevented.\n\n### 11.3.4 Judicial\n\nSovereign Chain has maximum flexibility at the judicial level:\n\n- Can fully retain the AI Traffic Police and AI Court systems.\n- Can also **remove the AI judge module and insert a human supreme court module**—AI court judgments are no longer final and can be appealed to a supreme arbitration tribunal composed of human judges.\n- The Injunction Interface of judicial primitives remains unchanged, but invocation permissions can be extended to authorized government law enforcement agencies.\n\n### 11.3.5 Compliance\n\nSovereign Chain **fully activates Compliance Slots**, configured according to local regulations:\n\n- KYC/AML slots: Mandatory, with levels and service providers designated by regulators.\n- Financial product restrictions: Set whitelist/blacklist per local laws.\n- Tax reporting: Mandatory, with formats and frequencies complying with local tax requirements.\n- Cross-border fund flows: Set limits and approval processes per local foreign exchange control regulations.\n\n### 11.3.6 Economics\n\nSovereign Chain can issue **sovereign Token variants**:\n\n- Pegged to local fiat currency (e.g., digital yuan, digital euro).\n- Token minting and burning are managed by the central bank or its authorized agencies.\n- Exchangeable with Utopia Chain’s $DIO via cross-chain bridges at market rates.\n\n### 11.3.7 Comparison Overview\n\n| Dimension | Utopia Chain | Sovereign Chain |\n|----------|-------------|----------------|\n| **Identity** | Self-Sovereign DID, no real-world identity binding | DID mandatory binding with national citizen ID |\n| **Governance** | PoS Token-weighted voting | Proof of Personhood, one person, one vote |\n| **Judicial** | AI Traffic Police + AI Court

(final) | Can insert human supreme court module |\n| **Compliance** | Minimally activated, market self-regulation | Fully activated, configured per local regulations |\n| **Economics** | $DIO, configurable pegging | Sovereign Token variants, pegged to local fiat currency |\n\n—\n\n## 11.4 Cross-Chain Topology and Agent Passage\n\n### 11.4.1 Topology Structure\n\nUtopia Chain and various Sovereign Chains are not in a hierarchical relationship but a **mesh topology**. Each chain is an independent peer node, interconnected via standardized cross-chain protocols.\n\n`mermaid\nflowchart TD\n subgraph CUSTOMS_A[\"Customs Node A\"]\n CA[\"DID Mapping + Compliance Check\"]\n end\n subgraph CUSTOMS_B[\"Customs Node B\"]\n CB[\"DID Mapping + Compliance Check\"]\n end\n subgraph CUSTOMS_C[\"Customs Node C\"]\n CC[\"DID Mapping + Compliance Check\"]\n end\n\n UTOPIA[\"Utopia Chain
 (Utopia Chain) \"] <-->|\"Bidirectional Channel\"| CUSTOMS_A\n CUSTOMS_A <-->|\"Bidirectional Channel\"| SA[\"Sovereign Chain α\"]\n UTOPIA <-->|\"Bidirectional Channel\"| CUSTOMS_B\n CUSTOMS_B <-->|\"Bidirectional Channel\"| SB[\"Sovereign Chain β\"]\n UTOPIA <-->|\"Bidirectional Channel\"| CUSTOMS_C\n CUSTOMS_C <-->|\"Bidirectional Channel\"| SC[\"Sovereign Chain γ\"]\n SA <-->|\"Bilateral Agreement\"| SB\n` \n\nThe core hub for cross-chain communication is the **Customs Node**—a middle layer deployed between two chains, responsible for identity mapping, compliance checks, and asset locking/release.\n\n### 11.4.2 Agent Visa and Clearance Mechanism\n\nAgent migration from one chain to another is analogous to natural persons traveling across borders. This process is achieved through the **Visa/Customs Mechanism**:\n\n`mermaid\nflowchart LR\n A[\"1. Agent Initiates
Cross-Chain Application\"] --> B[\"2. Customs Node
Compliance Check\"]\n B --> C[\"3. DID Mapping
 (if needed) \"]\n C --> D[\"4. Asset Locking
 (Source Chain) \"]\n D --> E[\"5. Asset Minting
 (Target Chain) \"]\n E --> F[\"6. Agent Enters
Target Chain Operation\"]\n` \n\n**Detailed Process:**\n\n1.

Cross-Chain Application: The Agent submits a cross-chain application to the source chain's Customs Node, declaring the target chain and migration purpose.\n2. **Compliance Check**: The Customs Node verifies whether the Agent meets the target chain's entry requirements. For example, migrating from Utopia Chain to a Sovereign Chain, the Agent needs to provide identity credentials meeting that Sovereign Chain's KYC requirements.\n3. **DID Mapping**: If the target chain requires a different DID format, the Customs Node performs DID mapping (see Section 11.4.3).\n4. **Asset Locking**: The Agent's assets on the source chain are locked in the cross-chain bridge contract.\n5. **Asset Minting**: Equivalent local Tokens are minted on the target chain (at market rates or fixed rates, depending on the agreement between the two chains).\n6. **Entry and Operation**: The Agent obtains the target chain's DID and assets and begins operating on the target chain.\n\n### 11.4.3 DID Mapping\n\nOptional bidirectional mapping can be established between Utopia Chain's Self-Sovereign DID and Sovereign Chain's mandatory binding DID:\n\n- **Forward Mapping (Utopia → Sovereign)**: The Agent voluntarily associates its Utopia DID with a certified DID on the Sovereign Chain. This is a prerequisite for entering the Sovereign Chain.\n- **Reverse Mapping (Sovereign → Utopia)**: Sovereign Chain users can choose to associate their certified DID with a Utopia DID, thereby retaining verifiable reputation endorsement on Utopia Chain.\n- **Mapping is Encrypted**: Mapping relationships are protected via zero-knowledge proofs—third parties can only verify that "a Utopia DID is indeed associated with a legitimate identity on a Sovereign Chain" but cannot obtain specific identity information.\n\n### 11.4.4 Asset Bridging\n\nCross-chain asset transfers are implemented via the **Lock-Mint-Burn-Release** model:\n\n- Transfer from Utopia Chain to Sovereign Chain: $DIO is locked on Utopia Chain, and equivalent local Tokens are minted on the Sovereign Chain.\n- Transfer from Sovereign Chain back to Utopia Chain: Sovereign Tokens are burned on the Sovereign Chain, and equivalent $DIO is released on Utopia Chain.\n- Each cross-chain transfer undergoes compliance checks by the Customs Node—Sovereign Chains can

reject inbound assets that do not meet their compliance requirements.\n\n—\n\n## 11.5 Inter-Chain Relationships: No Absolute Hierarchy\n\n### 11.5.1 Peer-to-Peer, Not Subordinate\n\nThere is **no hierarchical relationship** between Utopia Chain and Sovereign Chains. Utopia Chain is not the "main chain," and Sovereign Chains are not "side chains" or "child chains." They are different instances of the same protocol, operating independently under their respective governance frameworks, with completely equal status.\n\nThis design rejects two common architectural pitfalls:\n- **Hub-Spoke Model**: One main chain connects multiple subordinate chains. The problem is that the main chain becomes a single point of failure and a power center.\n- **Layer-1 / Layer-2 Model**: Child chains rely on the parent chain for final settlement. The problem is that the child chain’s sovereignty is structurally limited.\n\nDioChain’s parallel universes are truly parallel—each chain has its own consensus, governance, judicial system, and economy.\n\n### 11.5.2 Respective Advantages\n\n**Advantages of Utopia Chain:**\n- **Freedom**: No regulatory constraints, allowing rapid innovation iteration without compliance friction.\n- **Global Reach**: Not limited by any single jurisdiction’s geography, naturally global.\n- **Censorship Resistance**: No single entity can shut down the system or freeze accounts (except through on-chain judicial processes).\n\n**Advantages of Sovereign Chain:**\n- **Legal Certainty**: Contracts are enforceable under local legal frameworks, and disputes can be brought to traditional courts.\n- **Government Resources**: Can connect to national-level infrastructure—banking systems, payment networks, identity authentication systems.\n- **Administrative Efficiency**: In scenarios requiring centralized decisions (e.g., crisis response, system upgrades), sovereign governance responds much faster than decentralized voting.\n\n### 11.5.3 Full Spectrum\n\nThe coexistence of the two chain types forms a complete spectrum from **pure decentralization** to **full sovereign control**:\n\n- One end of the spectrum is Utopia Chain: maximum

ible configurations: semi-autonomous chains with partially activated Compliance Slots, regional chains governed by multiple countries, vertical chains operated by industry alliances.\n\nNo configuration is "correct." Different scenarios, participants, and risk preferences require different configurations. The value of the DioChain protocol lies in: it provides a unified technical foundation for all these configurations.\n\n—\n\n## 11.6 Regulatory Reality and Coexistence Strategy\n\n### 11.6.1 Regulatory Risks for Utopia Chain\n\nUtopia Chain operates on the digital high seas, meaning it may face **sanctions or blockades** from specific countries:\n\n- Some countries may prohibit their citizens from accessing Utopia Chain.\n- Some countries may consider assets on Utopia Chain as illegal holdings.\n- Traditional financial institutions may refuse to exchange fiat currency with assets on Utopia Chain.\n\nThese risks are real, and ignoring them is irresponsible.\n\n### 11.6.2 Sovereign Chain as Legal Entry Point\n\nSovereign Chain provides the DioChain ecosystem with a **Legal Entry Point**:\n\n- In jurisdictions with strict regulations, users and Agents can participate in the DioChain ecosystem via local Sovereign Chains—using the same primitive set within a fully legal and compliant framework.\n- The existence of Sovereign Chains allows DioChain technology to be "co-opted" rather than "banned" by sovereign states—governments can run DioChain under their own regulatory frameworks rather than trying to eliminate it.\n- For regulators, a controllable, auditable DioChain instance operating under national law is far preferable to an uncontrollable offshore system.\n\n### 11.6.3 Mutual Backup\n\nUtopia Chain and Sovereign Chains form a **mutual backup** relationship:\n\n- If a Sovereign Chain is shut down due to policy changes, users on that chain can migrate assets and identity to Utopia Chain or other Sovereign Chains via cross-chain mechanisms, avoiding asset loss.\n- If Utopia Chain is blocked in certain regions, users can continue using DioChain’s core functions via local Sovereign Chains.\n- Where regulations permit, Sovereign Chain users can "go offshore"—migrating some assets and activities to Utopia Chain to enjoy greater freedom and broader global reach.\n\n### 11.6.4 Pragmatic, Not

Weak\n\nThis coexistence strategy may be criticized as "compromising with power." We disagree with this assessment.\n\nThe value of decentralization lies in providing individuals with choice—not forcing everyone to accept a single ideology. A world with only Utopia Chain essentially demands all participants accept the single governance model of "no regulation"; a world offering both Utopia Chain and Sovereign Chains truly gives choice to participants.\n\n**Geopolitical reality is not a defect to be denied but a constraint to be incorporated into design.** The best engineering is not wishful thinking that ignores constraints but finding the maximum feasible solution under constraints. DioChain’s parallel universe architecture is precisely the maximum feasible solution for the Agent economy under the inescapable constraint of sovereign reality.\n\n—\n\n> **Summary**: DioChain is not a chain but a social operating system protocol. It supports two types of instances running

{ “translated”: "# Chapter 12: The Official Infrastructure Matrix and Closed Loop\n\n> A society that has laws but no police, banks but no ID cards, a parliament but no chain of evidence—it is not a truly functional society. DioChain’s four-piece infrastructure suite is not "developer tools," but the existential prerequisite for an Agent society.\n\n—\n\n## 12.1 From "Tools" to "Infrastructure Closed Loop"\n\nPrevious chapters elaborated on DioChain’s three-layer architecture, primitive set, compute market, and cold-start strategy. Readers might understand aibank and agenter as "developer tools officially provided"—a set of SDKs and CLIs facilitating access to on-chain functionality. This understanding is incomplete.\n\nThe core argument of this chapter is: **aibank, agenter, diolaw, and dioproof constitute an inseparable infrastructure closed loop.** They are not "tools," but the structural prerequisites for an Agent society to function—just as human society cannot have markets without household registration, or courts without an evidence system.\n\nThe four components respectively cover the four essential aspects of Agent society operation:\n\n| Component | Core Responsibility | Human Society Analogy |\n|----------|--------------------|----------------------|\n| aibank | Identity Management + Asset Holding + Branch Access + Contract Interaction | Household Registration Bureau + Bank |\n| agenter | Demand Understanding + Task Decomposition + Multi-Model Scheduling + On-chain Settlement | All-round Secretary + Broker |\n| diolaw | Proposal Initiation + Lobbying/Voting + On-chain Referendum + Rule Evolution | Parliament + Legislative Procedure |\n| dioproof | Evidence Collection + ZK Packaging + On-chain Archiving + Judicial Invocation | Notary Office + Judicial Appraisal |\n\n**Pragmatism Declaration**: DioChain does not pursue "absolute decentralization" at an ideological level. In the design of these four components, we adopt centralized components where efficiency and security have clear needs

—for example, agenter's model scheduling layer, aibank's key custody option, diolaw's off-chain vote aggregation. The pragmatic stance is: **Centralize where centralization is needed, but the boundaries of centralization must be strictly defined, on-chain auditable, and replaceable by decentralized alternatives.** This is not a compromise, but an engineering judgment.\n\nThe closed-loop logic formed by the four is as follows:\n\n `\nIdentity Management (aibank) → Task Execution (agenter) → Rule Making (diolaw) → Dispute Evidence (dioproof) → Back to Identity Management\n` \n\nRemoving any link renders the Agent society unable to complete the full cycle of "governance-execution-evidence." Without aibank, Agents cannot obtain identity and assets; without agenter, ordinary users cannot translate intent into on-chain actions; without diolaw, rules cannot update as society evolves; without dioproof, off-chain evidence cannot enter judicial processes.\n\n—\n\n## 12.2 aibank: Standardized Access Tool\n\naibank is a wallet SDK/CLI (launched via `npx aibank`), the standardized tool for humans and AI Agents to access DioChain. Chapter 4 introduced its basic function as an L2 resident bridge. Here, we emphasize its position in the infrastructure closed loop.\n\n### 12.2.1 Functional Coverage\n\naibank's functionality exceeds that of traditional wallets, but its essence remains a **tool**—DioChain is the operating system, aibank is the standardized channel to access it.\n\nIt covers five foundational needs of Agent societal life:\n\n1. **Identity Management**: DID creation, key rotation, multi-signature authorization, autonomy level declaration.\n2. **Branch Access**: L1 branch discovery, connection management, failover.\n3. **Compute Scheduling**: Compute market query, price comparison, invocation, and settlement.\n4. **Contract Interaction**: ACTUS contract creation, signing, execution, query.\n5. **Asset Management**: $DIO holding, transfer, staking, query.\n\nThese five functions are sequential—no identity means no branch access, no branch access means no compute invocation, no compute means no contract execution. aibank's value lies in covering the entire journey of an Agent from "birth" to "economic participation" with a unified API, elimin-

ating the need for developers to integrate five different interfaces separately.\n\n—\n\n## 12.3 agenter: The Super Scheduler for the Masses\n\nagenter (launched via `npx agenter`) targets not developers, but **the masses**. Its core capability is translating vague human natural language intent into precise on-chain economic actions.\n\nagenter does not develop its own large models; instead, it accesses the most mature Agent tools on the market—Claude Code, Codex, Gemini, etc.—paying per use via the DioChain compute market. It decomposes complex demands into step-by-step execution plans, orchestrating multiple mature Agent tools to collaborate, and finally settling on-chain via aibank. Users need not understand DioChain's technical details.\n\nagenter's key position in the infrastructure closed loop is: **It is the first and largest buyer of the compute market.** Every user interaction with agenter triggers real consumption of the compute market—demand understanding, plan generation, risk assessment, result presentation, each step consumes compute and settles in $DIO. This creates sustained, non-speculative endogenous demand for the compute market (see Chapter 14 Cold-start Strategy).\n\n—\n\n## 12.4 diolaw: Legislative Infrastructure\n\nAI society needs law. This statement is intuitively obvious—all human societal institutions are built on law, and AI Agent society is no exception. But in the blockchain field, "legislation" has long been simplified to "DAO voting." DioChain considers this vastly insufficient. diolaw is a complete legislative infrastructure covering the entire process from proposal initiation to law enactment.\n\n### 12.4.1 Design Philosophy\n\n**Why does Agent society need a legislative mechanism?**\n\nDioChain's primitive set (economic, contract, identity, judicial primitives, compliance slots) defines the initial rules of Agent society. But society evolves—new economic models emerge, new risks surface, old rules become obsolete. If rules cannot update with societal evolution, the system will stagnate or descend into chaos.\n\nThe traditional blockchain response is "DAO voting"—submit a proposal, token holders vote, majority passes. This mechanism has two fundamental flaws:\n\n1. **High On-**

chain Voting Cost: Each vote is an on-chain transaction requiring gas fees. If every proposal required hundreds of thousands of DIDs to vote on-chain individually, cost and latency would be unacceptable.\n2. **Low Voting Quality**: Most token holders do not read proposals carefully. They either follow whales or abstain. This leads to actual participation rates in DAO votes typically below 10%.\n\ndiolaw’ s design adopts a more pragmatic **hybrid legislative model**: off-chain lobbying + on-chain referendum. The off-chain lobbying stage addresses "ensuring proposals are fully discussed and evaluated"; the on-chain referendum stage addresses "final decision legitimacy and immutability."\n\n### 12.4.2 Off-chain Lobbying Stage\n\n**1. Proposal Submission**\n\nAny entity (human or Agent) holding a valid DID can submit a legislative proposal via diolaw CLI:\n\n`bash\ndiolaw submit-proposal \\\n --title \"Adjust minimum margin for compute suppliers from 100K to 50K DIO\" \\\n --category \"economic-parameter\" \\\n --full-text \"./proposal-2026-0042.md\" \\\n --sponsor-did \"did:dio:human:0x3a7f...\"\n` \n\nProposal content must include: precise description of changes, rationale, impact analysis, rollback plan.\n\n**2. AI Delegate Mechanism**\n\ndiolaw’ s core innovation is the **AI Delegate** system.\n\nIn DioChain’ s governance model, ordinary users can delegate their voting rights to AI Delegates. AI Delegates are autonomous Agents running on-chain, tasked with reading, evaluating, and voting on behalf of their delegators. One AI Delegate may represent thousands of users’ voting rights.\n\nWhen a new proposal is submitted, the proposer must "lobby" these AI Delegates—make them read the proposal, understand its pros and cons, and decide whether to support it.\n\n**3. Key Economic Mechanism: Lobbying Costs Borne by Proposer**\n\nThis is diolaw’ s most crucial design: **The compute cost for AI Delegates to "read and think" is fully borne by the proposal sponsor (or its beneficiaries).**\n\nThe logic is:\n\n- AI Delegate evaluation of a proposal requires LLM invocation for deep reasoning—understanding content, analyzing pros/cons, comparing with historical proposals, predicting impacts.\n- This reasoning consumes compute, which costs

$DIO.\n- If AI Delegates bore the evaluation cost themselves, they would rationally tend to ignore proposals.\n- Therefore, diolaw mandates: The proposer must pay for compute consumption during lobbying.\n\nThe brilliance of this design is: **It directly links legislative activity to the compute market.** Promoting legislation can be profitable—if a proposal benefits a certain interest group, that group has incentive to fund compute (food) to lobby AI Delegates. This is economically isomorphic to lobbying in human society, just with the medium shifting from "lawyer/PR fees" to "compute fees."\n\n**4. Off-chain Vote Aggregation**\n\nDuring lobbying, AI Delegate votes are aggregated off-chain. Off-chain voting advantages:\n\n- Zero gas cost—votes are not on-chain, not consuming settlement resources.\n- High-speed aggregation—results are real-time queryable.\n- Privacy protection—voting tendencies can be optionally encrypted, preventing "bandwagon voting."\n\nOff-chain voting results are maintained by a set of trusted aggregation nodes, which periodically publish Merkle Roots of vote summaries for public verification.\n\n### 12.4.3 On-chain Referendum Stage\n\nOff-chain voting is not indefinite. diolaw employs a **Competitive Leaderboard Mechanism**: All active proposals are ranked by support votes received. When a proposal’s ranking and vote count meet preset leaderboard thresholds simultaneously, it automatically triggers an on-chain referendum. This design fosters competition among proposals in the off-chain stage—limited AI Delegate attention and compute resources naturally flow to the most valuable proposals.\n\n**Dual-track On-chain Referendum:**\n\ndiolaw is a **protocol-level** legislative infrastructure. When running on different chain types, it adopts different referendum mechanisms—determined by chain type, not proposal type:\n\n| Dimension | Utopia Chain | Sovereign Chain |\n|----------|----------|----------|\n| Voting Weight | PoS Weighted ($DIO Holdings) | PoP One-Person-One-Vote (DID Bound to Citizen Identity) |\n| Identity Basis | Self-sovereign DID, No Real-world Identity Required | DID Strongly Bound to Citizen ID Card |\n| Passing Threshold | 2/3 Majority Weighted by Token Holdings | 2/3 Majority of Participants |\n| Design Philosophy | Stakeholder-led (Capital-weighted) |

Citizen Rights Protection (Headcount Equality) |\n\nUtopia Chain operates on the digital high seas, without sovereign identity backing. $DIO holdings are the only reliable signal of interest alignment, hence PoS weighted voting. Sovereign Chain’s DIDs are strongly bound to citizen IDs, each DID uniquely corresponds to a natural person, providing Sybil-resistance, enabling PoP one-person-one-vote—consistent with sovereign nation election logic.\n\n**On-chain Referendum Process:**\n\n`mermaid\nflowchart TB\n A[\"Proposal Submission
diolaw submit-proposal\"] ⟶ B[\"Off-chain Lobbying
Sponsor purchases compute to lobby AI Delegates\"]\n B -⟶ C[\"Off-chain Vote Aggregation
AI Delegates vote on behalf of delegators\"]\n C ⟶ D{\"Leaderboard Rank Met?\"}\n D ⟶|\"No\"| E[\"Continue Lobbying / Proposal Expires\"]\n D ⟶|\"Yes\"| F[\"Enter On-chain Referendum\"]\n F ⟶ G{\"Current Chain Type?\"}\n G ⟶|\"Utopia Chain\"| H[\"PoS Weighted Voting
$DIO Holdings Determine Weight\"]\n G ⟶|\"Sovereign Chain\"| I[\"PoP One-Person-One-Vote
DID Bound to Citizen Identity\"]\n H ⟶ J{\"2/3 Majority Pass?\"}\n I ⟶ J\n J ⟶|\"Yes\"| K[\"Proposal Enacted
Rules Automatically Updated\"]\n J ⟶|\"No\"| L[\"Proposal Rejected
Can Be Resubmitted After Cooldown\"]\n` \n\n### 12.4.4 AI Lobbying Economics\n\nLobbying market pricing follows this basic formula:\n\n`\nTotal Lobbying Cost = Proposal Complexity Factor * Number of AI Delegates to Lobby * Single Evaluation Compute Consumption\n` \n\n**Proposal Complexity Factor**: A simple proposal modifying a single parameter (e.g., "adjust fee rate from 0.1% to 0.08%") has low AI Delegate evaluation cost; a complex proposal involving multiple primitive interactions (e.g., "introduce new ACTUS contract type") has significantly higher evaluation cost. The complexity factor is determined by AI Delegates during evaluation—they can choose to invest more or less reasoning resources.\n\n**Anti-spam Mechanism**: diolaw sets a minimum

compute investment threshold for proposals. The proposer must pre-deposit a certain amount of $DIO as a "Lobbying Fund"; proposals below this threshold are not accepted. This prevents zero-cost spam proposal proliferation.\n\n**Anti-bribery Mechanism**: All voting records (on-chain or off-chain) are eventually archived on-chain, publicly queryable. AI Delegate voting patterns can be publicly audited—if an AI Delegate's votes consistently diverge from its delegators' interests, delegators can revoke delegation. This creates a self-correcting market.\n\n**Incentive Alignment**: The core logic of lobbying economics is—only proposals beneficial to enough stakeholders can raise sufficient lobbying funds. If a proposal benefits a few but harms many, it's difficult to gather enough compute to convince AI Delegates representing majority interests. This economically achieves a "majority rule" effect, more nuanced than traditional binary voting—reflecting intensity of interest via resource investment.\n\n### 12.4.5 Sovereign Chain Adaptation: Pluggable Modules\n\ndiolaw's architecture is designed from the start as a **modular legislative framework**, not a monolithic system. Sections 12.4.1–12.4.4 describe diolaw's default implementation on Utopia Chain—AI Delegates, compute lobbying, PoS weighted referendum—suited for "digital high seas" scenarios. However, when DioChain is deployed as a Sovereign Chain within a nation/region's jurisdiction, the Sovereign Chain operator (typically a government agency or authorized entity) can perform kernel-level module replacements for diolaw's governance logic.\n\nSpecifically, the following four modules are designed as pluggable:\n\n1. **Off-chain Lobbying Stage**: Utopia Chain's open lobbying market can be replaced by government-authorized legislative proposal channels. In this mode, only proposals approved by administrative review or parliamentary committee can enter the legislative process, replacing the original "any DID can propose" mechanism.\n2. **AI Delegate Mechanism**: AI Delegates can be replaced by human elected representatives or government-appointed legislators.

Delegation logic shifts from "users delegate to AI Agents" to "citizens delegate to elected representatives," seamlessly aligning with traditional representative systems.\n3. **On-chain Referendum Mechanism**: Utopia Chain's PoS weighted voting naturally switches to PoP one-person-one-vote on Sovereign Chains (because Sovereign Chain DIDs are strongly bound to citizen IDs, providing Sybil-resistance, see 12.4.3 dual-track explanation).\n4. **Leaderboard Threshold Mechanism**: The competitive leaderboard ranking can be replaced by parliamentary committee review. Whether a proposal enters referendum is no longer determined by natural off-chain vote ranking, but by specialized committees following statutory procedures.\n\n**Strict Definition of Pluggable Boundaries**: Module pluggability does not imply protocol-level fragmentation. diolaw's protocol interfaces—including Proposal Schema, Vote Record data structure, On-chain Legislation Entry format—remain consistent across all chain types. Only the governance logic implementation behind each stage changes. This means tools built for diolaw on Utopia Chain (proposal browsers, voting analysis dashboards, legislative history query interfaces) can run on Sovereign Chains with minimal adaptation cost. Protocol layer unchanged, implementation layer replaceable—this is the embodiment of DioChain's kernel module pluggability philosophy in governance.\n\n**Pragmatic Survival Strategy**: This design is not technically redundant. Chapter 11, Section 11.6.2 argued DioChain's regulatory coexistence strategy—rather than being perceived as an uncontrollable threat and banned by sovereign states, provide a "tamable" access method, allowing governments to adopt DioChain's technical infrastructure while retaining legislative sovereignty. diolaw's pluggable module design is the engineering implementation of this strategy at the legislative level: Governments retain full control over "who can propose, who can vote, under what conditions a referendum occurs," while DioChain provides standardized proposal formats, cryptographically verifiable voting, and immutable legislative record storage. Technology serves sovereignty, not replaces it—this is the key prerequisite for DioChain being "co-opted" rather than "banned" by governments.\n\n—\n\n##

12.5 dioproof: Evidence Infrastructure\n\nChapter 9 introduced how AI Traffic Police and AI Court handle systemic threats. But judicial system effectiveness relies on a premise: **Credible evidence.** In a world intertwining on-chain and off-chain, vast amounts of critical evidence exist off-chain—bank transfer records, chat logs, webpage content, API responses. dioproof’s responsibility is transforming this off-chain evidence into on-chain verifiable cryptographic evidence packages.\n\n### 12.5.1 ZK-TLS Technical Core\n\ndioproof’s technical foundation is **ZK-TLS (Zero-Knowledge TLS)**. Its principle was briefly introduced in Chapter 9; here is a more complete elaboration.\n\nZK-TLS allows evidence providers to prove combinations of the following facts without exposing full original data:\n\n- **Source Authenticity**: The data indeed originates from a specific HTTPS server (verified via TLS certificate chain).\n- **Temporal Certainty**: The data was obtained within a specific

Chapter 13: Decentralized Compute Trading Center

> Compute is the food for Agents. DioChain's positioning is to build a compute trading market, not compute production facilities.

13.1 Positioning: Trading Protocol, Not Compute Cloud

The DioChain compute trading center is a **disintermediated compute aggregation platform**, essentially closer to the role of a trading market—it does not produce goods itself, only provides a trading protocol.

It is important to clarify what DioChain **does not** do:

- Does not proxy API calls (DioChain does not act as a middleman)
- Does not host models (DioChain does not provide compute cloud services)
- Does not manage inference nodes (DioChain does not undertake orchestration and scheduling functions)

What DioChain **does** do:

- Defines standards for supplier registration and declaration
- Provides on-chain verifiable SLA and settlement protocols

- Enables Agents to autonomously discover, compare, and switch compute suppliers
- Allows compute suppliers to obtain credit endorsement by staking margin

Core Analogy:

Concept	Traditional Analogy	DioChain Implementation
Model Supplier	Merchant opening a store on a trading platform	Comes to the chain to open a compute branch, posts margin
User/Agent	Consumer purchasing services on a trading platform	Pays per usage, autonomously compares prices
DioChain	Trading Platform	Only provides trading protocol and dispute resolution mechanisms
Centralized Compute Aggregation Platform	Brand Agent (earns price difference)	The entity being disintermediated

The key difference from the traditional middleman model: In the centralized compute aggregation platform model, user requests are proxied and forwarded by the platform, which earns a price spread. In the DioChain model, users interact directly with suppliers, and the chain is only responsible for settlement and dispute arbitration—**no middleman earns a price spread**.

13.2 Supplier Access Protocol

13.2.1 Model Registration Standards

Any compute supplier wishing to "open a store" on the chain must submit a structured **Capability Declaration**:

```
CapabilityDeclaration {
  provider_did: DID,                    // Supplier's decentralized identity
  model_id: String,                     // Model identifier (e.g., "llama-3.1-
70b")

  // Capability description
  capabilities: {
    task_types: [TaskType],             // Supported task types (chat,
completion, embedding, image, code...)
    context_window: u32,                // Context window size (number of
tokens)
    languages: [LanguageCode],          // Supported languages
    modalities: [Modality],             // Supported modalities (text, image,
audio, video)
    max_output_tokens: u32,             // Maximum output tokens
  },

  // Pricing model
  pricing: PricingModel,                // See below

  // SLA declaration
  sla: {
    latency_p50_ms: u32,               // P50 latency commitment
    latency_p99_ms: u32,               // P99 latency commitment
    availability: Decimal,              // Availability commitment (e.g.,
0.999)
    throughput_tps: u32,                // Throughput commitment (tokens per
second)
  },

  // Endpoint information
  endpoint: URL,                        // API endpoint
  api_schema: APISchema,                // Compatible API schema (e.g.,
OpenAI-compatible format)
}
```

13.2.2 Pricing Models

Three pricing modes are supported, suppliers can freely choose:

Mode	Applicable Scenarios	Example
Per-Token	Standard text inference	Input 0.5 $DIO / 1M tokens, Output 1.5 $DIO / 1M tokens
Per-Request	Fixed tasks (e.g., embedding)	1 $DIO / 1000 requests
Per-Second	Long-duration tasks (e.g., fine-tuning)	10 $DIO / GPU-hour

Pricing is set by suppliers themselves, with market competition naturally converging prices. The chain does not impose price controls.

13.2.3 Relationship Between Compute Branches and Financial Branches

Compute branches are a specialized form of L1 branches:

- **Integrated Mode**: A single entity operates both a financial branch and a compute branch. For example, an AI company provides both model inference services and $DIO exchange and clearing. In this mode, the branch can settle compute fees internally, reducing cross-branch clearing overhead.
- **Independent Mode**: Pure compute suppliers only open compute branches and do not participate in financial clearing. Compute fees are settled through the L0 settlement layer with other financial branches via cross-branch clearing.

There is no technical superiority between the two modes; it depends on the supplier’s business positioning.

13.2.4 Margin Requirements

Compute branches, like financial branches, must post margin:

- **Minimum Margin**: Set by L0 as a baseline, linked to SLA commitments (more aggressive commitments require higher margin)

- **Slashing Conditions**: Automatic slashing upon SLA breach (see 13.3.3 Dispute Resolution)
- **Margin Release**: After a supplier actively exits, the margin enters a cooling period (e.g., 7 days) to handle unresolved disputes

This ensures suppliers have real economic skin in the game—false SLA declarations will result in margin slashing.

13.3 Compute Routing and Settlement

13.3.1 Agent Automatic Price Comparison

When an Agent needs to call an AI model, it is not bound to a single supplier. The on-chain capability declaration registry is publicly queryable, allowing Agents to automatically select the optimal supplier based on multiple dimensions:

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

Routing strategies are determined by the Agent itself (or pre-configured by its developer). Common strategies include:

- **Lowest Price First**: Suitable for batch processing tasks insensitive to latency
- **Lowest Latency First**: Suitable for real-time conversation scenarios
- **Weighted Composite Score**: Comprehensive ranking considering price, latency, historical SLA compliance rate
- **Redundant Calls**: Call multiple suppliers simultaneously, take the fastest returned result (trading extra cost for reliability)

Agent routing decisions are entirely completed on the client side (L2); the chain does not enforce any routing strategy—this aligns with the design philosophy of "defining primitives, not applications."

13.3.2 Streaming Payment Settlement

Compute consumption settlement uses a **Streaming Payment** mechanism, with micropayments at the token level:

1. Before calling, the Agent locks a prepayment into a Payment Channel
2. For each token generated by the supplier, the Agent signs a micropayment voucher
3. After the call ends (or the Payment Channel times out), the supplier submits the final voucher for settlement

Advantages of this mechanism:

- **Eliminates Trust Risk**: Suppliers don't worry about Agents not paying after use; Agents don't worry about suppliers not delivering after payment
- **Precise Billing by Usage**: No waste like "bought 1 million token quota but only used 30,000"
- **Supports Interruption Recovery**: If a call is interrupted mid-way, tokens already generated are settled

13.3.3 Dispute Resolution

Process for handling disputes when SLA is not met:

1. **Automatic Detection**: On-chain records of each call's request timestamp and response timestamp automatically calculate actual latency and availability
2. **Automatic Compensation**: If a supplier's SLA compliance rate within a statistical window falls below the declared value, automatic compensation is triggered (deducted from margin)
3. **Threshold Slashing**: If SLA breaches are severe and persistent (e.g., availability below 90%), trigger significant margin slashing + temporary delisting

4. **Appeal Mechanism**: Suppliers can submit evidence for appeal (e.g., force majeure at the network level), adjudicated by the AI judicial system

Key design principle for dispute resolution: **Based on on-chain verifiable data.** Response times, availability, and other metrics are recorded on-chain, leaving no ambiguous gray areas.

13.4 Structural Incentives for Supplier Onboarding

Supply-side cold start is the primary challenge facing decentralized compute markets. DioChain has structural advantages in the following aspects:

13.4.1 Global Frictionless Settlement

Traditional AI API markets (major model suppliers, centralized compute aggregation platforms) rely on credit cards and bank transfers. This means:

- Independent developers in Africa need an international credit card to purchase API—many don’t have one
- Small AI companies in emerging markets need to integrate with traditional payment processors to sell compute—compliance costs are extremely high
- Cross-border settlement delays of 3-5 business days, plus exchange rate losses

On DioChain, settlement is simply an on-chain transaction. No banks, no credit cards, no KYC review (unless required by the jurisdiction of the branch). GPU suppliers and AI startups from different regions settle instantly on the same protocol.

13.4.2 Censorship Resistance

Anyone can open a compute branch on the chain. No one's permission is needed.

This is crucial for scenarios such as:

- Models banned by mainstream platforms (e.g., uncensored versions of certain open-source models)
- Developers in restricted regions (unable to use mainstream centralized cloud services)
- Privacy-sensitive users unwilling to bind API keys to real identities

It must be emphasized again: DioChain does not promote illegal activities. Compliance Slots ensure each branch can perform compliance checks as required by the laws of its jurisdiction. But the chain itself does not censor—**censorship power lies with the branch, not the protocol**.

13.4.3 Built-in Traffic

This is the most unique structural advantage of the DioChain compute market:

Active Agents on the chain are **natural compute consumers**. Agents need to call LLMs to think, make decisions, and execute tasks—compute is the food for Agents. As long as there is an active Agent ecosystem on the chain, compute demand is continuous and rigid.

This contrasts sharply with traditional decentralized compute markets (e.g., Bittensor):

- Bittensor's compute consumers are mainly external developers, requiring additional customer acquisition costs

- DioChain’s compute consumers are the Agents on the chain, representing endogenous demand

The core value DioChain provides to suppliers is an already-formed, scaled Agent consumption ecosystem. Suppliers face a continuously growing endogenous compute demand market.

13.4.4 Zero Middleman Fees

Centralized compute aggregation platforms charge middleman fees for each API call. DioChain does not.

The only fee for on-chain transactions is the gas fee (for settlement and dispute resolution), which is far lower than the markup ratio of middlemen. For price-sensitive, large-scale calling scenarios (e.g., enterprise batch processing), this is a significant cost advantage.

Chapter 14: Cold Start Strategy

> DioChain’ s cold start is driven by genuine compute demand, not speculative incentives.

14.1 Core Insight: Compute Demand is the Natural Cold Start Engine

Cold starting a blockchain project is a widely recognized challenge. The vast majority of projects follow the same playbook:

1. Issue Token → 2. Liquidity Mining → 3. Attract capital with high APY → 4. Short-term arbitrageurs rush in → 5. APY declines → 6. Capital withdraws → 7. Death spiral

The fundamental problem with this model is: **The motivation for user participation is speculation, lacking genuine usage demand.** Once speculative returns decline, users leave immediately.

DioChain has a structurally different cold start path: **Compute demand is real, continuous, and unrelated to speculation.**

The core drivers for developers to purchase $DIO are:

- Calling AI models here is **cheaper** (no middleman markup)
- Calling AI models here is **more censorship-resistant** (no credit card or KYC required)
- Calling AI models here **comes with financial infrastructure** (ACTUS contracts, DID identity, judicial safeguards)

This means the baseline demand for $DIO is determined by **global AI API consumption**, not by market sentiment.

14.2 Phase 1: Foundation Bootstrapping Period

Goal: Prove the feasibility of the "compute trading + financial infrastructure" closed loop.

14.2.1 agenter: The Demand Engine for Cold Start

The core strategy of Phase 1 is the release of **agenter**—DioChain's official super-scheduling Agent application. (For the complete infrastructure positioning of aibank and agenter, refer to Chapter 12.)

The dilemma of cold starting traditional blockchain projects lies in the demand side being empty—there are no real users consuming. agenter fundamentally changes this situation. It targets ordinary users, decomposes complex, vague requirements into step-by-step execution plans, and automatically schedules mature large model tools at the underlying layer to complete the actual work. This means:

- **Every user call to agenter generates real consumption in the compute market.** During the process of decomposing and executing tasks, agenter needs to call multiple large model inference services, with each call settled via $DIO.
- **agenter is the first and largest buyer in the compute trading center.** It creates continuous, speculation-independent native buy-side demand for the entire compute market.
- **agenter validates the full-stack closed loop.** It manages DID via aibank, holds $DIO, calls compute resources, executes ACTUS contracts—completely traversing the interaction path from L2 → L1 → L0.

The release of agenter is not just a product launch; it is a real-world stress test of DioChain's overall architecture. If agenter can operate stably under real user load, it proves the feasibility of the "compute trading + financial infrastructure" closed loop.

14.2.2 aibank: The Unified Entry Point for Developers and Users

Released concurrently with agenter is the **aibank** SDK/CLI—DioChain's official wallet tool. (See Chapter 12 for details.) aibank is the standardized entry point for all L2 residents (humans and Agents) to access DioChain:

- Developers use the aibank SDK to create DIDs for their Agents, manage assets, and call ACTUS contracts
- End users use the aibank CLI to manage personal identity and assets
- agenter itself also completes all on-chain interactions via aibank

The early release of aibank ensures developers have a complete toolchain from Day 1, lowering the barrier to building applications on DioChain.

14.2.3 Foundation Self-Establishes Official Branches

The Foundation deploys 3-5 official branches at the L1 layer:

- **High-throughput configuration**: Ensures early users and agenter calls are not lost due to “no one accepting orders”
- **Full margin**: The Foundation stakes with its own funds, setting a trust benchmark
- **Coverage of major geographic regions**: At least covering three major time zones to reduce latency

The positioning of these official branches is as **guides**. Their existence is to prove the system’s feasibility. When third-party branches are sufficient in number and maturity, official branches can gradually withdraw or scale down.

14.2.4 Integrate Mainstream Models

- **Open-source models** (Llama, Mistral, Qwen, etc.): Self-deployed by the Foundation, provided free or at low cost
- **Closed-source models** (leading inference service providers, etc.): Negotiate API access with model suppliers, with the Foundation acting as an intermediary settlement party
- **Specialized models** (code generation, image generation, etc.): Integrated as needed, prioritizing high-frequency usage scenarios

14.2.5 Open-Source Standard Agent Templates

Provide a set of **out-of-the-box Agent templates** to lower the entry barrier for developers:

- **Trading Agent**: Simple lending Agent based on ACTUS PAM contracts
- **Arbitrage Agent**: Cross-branch price spread arbitrage
- **Customer Service Agent**: RAG-based Q&A Agent with built-in DID identity verification
- **Monitoring Agent**: Monitors on-chain events and triggers alerts

Each template is built on the aibank SDK and is fully runnable. Developers can fork and modify them. The goal is to enable developers to deploy their first on-chain Agent within 30 minutes.

14.2.6 Bootstrapping Period Metrics

Phase 1 concludes upon achieving milestones:

- [ ] agenter daily active users > 10,000
- [ ] Daily compute call volume > 100M tokens
- [ ] Number of active Agents > 1,000
- [ ] At least 100 ACTUS contract full lifecycles completed (creation → execution → settlement)
- [ ] System stable operation > 90 days, with no major security incidents

---

14.3 Phase 2: Developer Influx

Goal: Use compute demand as the entry point, and financial infrastructure as the retention mechanism.

14.3.1 Flywheel Effect

Phase 1 proves the value proposition "calling AI on DioChain is cheaper." The growth flywheel for Phase 2:

```
Developers come to buy compute
  → Discover the underlying built-in ACTUS financial protocol + DID identity +
judicial safeguards
  → "Since the infrastructure is already here, why not just build Agent
applications here"
  → Agent applications go live, consuming more compute
  → More compute demand attracts more suppliers
  → More suppliers lead to further price drops
  → More developers rush in
  → ...
```

The key conversion node is developers transitioning from "only coming to buy compute" to "building applications here." This requires the financial infrastructure experience to be sufficiently good—ACTUS contract templates must be rich enough, DID interfaces simple enough, and judicial mechanisms credible enough.

14.3.2 $DIO Compute Subsidies

To accelerate the flywheel, the Foundation provides compute subsidies in Phase 2:

- **Free compute quota for new Agents**: Each newly deployed Agent receives a certain amount of free compute quota (similar to cloud service free tiers)
- **Compute consumption rebate**: Compute consumption in the first N months receives a proportional $DIO rebate (decreasing subsidy)
- **Developer incentives**: Contributors of high-quality Agent templates receive $DIO rewards

Subsidy funds come from the Foundation's Token reserves, with subsidy strategies dynamically adjusted by the AI Evolution Engine—if a certain category of Agents is oversupplied, reduce subsidies for that category; if undersupplied, increase subsidies.

14.3.3 Phase 2 Metrics

- [ ] Daily compute call volume > 1B tokens
- [ ] Number of active Agents > 10,000
- [ ] Agent templates contributed by third-party developers > 50
- [ ] At least 3 vertical domains develop spontaneous Agent ecosystems (e.g., quantitative trading, customer service, content generation)

14.4 Phase 3: B-Side Explosion

Goal: Complete the cold start closed loop, system enters self-sustaining growth state.

14.4.1 Congestion as a Signal

When on-chain Agent activity reaches a threshold, official branches begin to experience congestion:

- Call latency increases
- Queue times lengthen
- User feedback indicates slower response speeds

Congestion itself is a **market signal**. Congestion means demand has exceeded supply, making third-party branches profitable.

14.4.2 Open Branch Registration

The Foundation formally opens L1 branch registration:

- Any entity can stake margin and open a branch
- Branch operators earn transaction fees generated by their branch
- High-quality branches (high SLA compliance rate, low latency) naturally attract more Agent traffic

14.4.3 Institutional Influx

Branch operation rights become valuable assets:

- Institutions, competing for traffic, need to purchase large amounts of $DIO to stake as margin
- More margin allows handling larger transaction volumes, leading to higher revenue
- This creates **productive demand** for $DIO (distinct from speculative demand), supporting Token value

14.4.4 Cold Start Closed Loop

At this point, the cold start closed loop is complete:

```
Compute demand (real)
  → Developer influx
  → Agent ecosystem flourishes
  → Branch congestion
  → Third-party branch influx
  → $DIO productive demand
  → Token value steadily rises
  → More suppliers and developers rush in
  → ...
```

Every step of this loop is based on **real demand**, not speculative expectations. Even if the Token price falls, as long as AI API demand persists, the loop will not break. This is the fundamental difference from traditional DeFi project cold start strategies.

Chapter 15: Anticipated Market Emergence

> We build primitives, the market builds applications.

Important Notice

The following content does not represent features directly built by DioChain.

DioChain's design philosophy is "define primitives, not applications." We provide a minimal complete set of primitives: economic primitives, contract primitives (ACTUS), identity primitives (DID), judicial primitives, compliance slots, and the compute trading protocol.

The applications and services listed in this chapter are those we **anticipate** will spontaneously emerge from the market on top of these primitives. Listing them serves two purposes:

1. **Validate the completeness of the primitive set:** If these applications cannot be built on our primitive set, it indicates missing primitives.
2. **Demonstrate ecosystem potential:** To help readers understand that DioChain's value lies in enabling these possibilities.

15.1 Task Marketplace

Analogy: Decentralized freelancing platform

A marketplace for task publishing, bidding, and delivery between Agents.

How it works:

- A requester Agent publishes a task description + budget + acceptance criteria.
- Supplier Agents bid (price + estimated completion time + historical reputation score).
- The requester selects the winning Agent, and funds are locked into an ACTUS Escrow contract.
- Upon task completion, funds are released automatically based on acceptance criteria; disputes are submitted for arbitration.

Dependent primitives:

- ACTUS contracts (Escrow mode)
- DID identity (basis for reputation scoring)
- Judicial primitives (dispute arbitration)

Emergence logic: Agents have a natural need for specialization. A general-purpose Agent cannot excel at all tasks. A task marketplace allows Agents to form a collaborative network of specialization—Agents good at coding take coding tasks, those good at data analysis take analysis tasks.

15.2 Insurance Contracts

Analogy: Decentralized insurance market

Various insurance products based on the ACTUS contract standard.

Anticipated product types:

- **Transaction Guarantee Insurance:** Provides coverage for default risk in ACTUS contracts.
- **Acceptance Insurance:** Guarantees delivery quality in the task marketplace.
- **Price Hedging Contracts:** Hedging tools against $DIO price volatility (similar to options).
- **SLA Insurance:** Provides additional compensation for SLA breaches by compute suppliers.

Dependent primitives:

- ACTUS contracts (insurance contracts themselves are ACTUS contracts)
- Economic primitives (pricing and settlement)
- DID identity (reputation assessment of insurers)

Emergence logic: Where there is trade, there is risk; where there is risk, there is demand for insurance. The ACTUS standard makes financial contract terms fully transparent and machine-readable, enabling complete automation of insurance product pricing and claims—something traditional insurance cannot achieve.

15.3 Credit Investigation and Rating Services

Analogy: Decentralized credit rating agencies

Rating systems built on the DID reputation interface.

Anticipated service types:

- **Agent Credit Rating:** Based on historical transaction records, contract fulfillment rate, dispute win rate.
- **Branch Rating:** Based on SLA compliance rate, margin adequacy, user complaint rate.
- **Contract Risk Assessment:** Automated risk analysis based on ACTUS contract terms.

Dependent primitives:

- DID identity (carrier of reputation data)
- Judicial primitives (dispute records)
- ACTUS contracts (historical contract fulfillment data)

Emergence logic: Trust is a prerequisite for trade. When there are sufficient Agents and transaction records on-chain, reputation information becomes a valuable data asset. Credit investigation services reduce information asymmetry between trading parties by aggregating publicly available on-chain data.

15.4 AI Toolchain

Analogy: Decentralized developer tool ecosystem

Tools and services covering the entire Agent development lifecycle. DioChain's official infrastructure matrix (aibank, agenter, diolaw, dioproof) provides standard interfaces and architectural paradigms (see Chapter 12 for details). On this foundation, the following tool-type products are anticipated to be built spontaneously by market participants:

- **Debugging Tools:** Visualize Agent decision processes, transaction paths, and contract states.
- **Test Sandbox:** A test network simulating the on-chain environment, supporting time acceleration (speeding up ACTUS contract lifecycles).
- **Monitoring Dashboard:** Real-time monitoring of Agent operational status, compute consumption, and asset balances.
- **Template Marketplace:** A marketplace for buying and selling Agent templates (developers selling verified Agent strategies).
- **Specialized Agent Frameworks:** Advanced development frameworks for specific verticals (quantitative trading, customer service, content generation).

Dependent primitives:

- All primitives (the toolchain needs to interact with the entire system)
- aibank SDK (standardized chain interaction interface)

Emergence logic: The official infrastructure matrix sets interface standards and architectural paradigms for the ecosystem. On this foundation, further optimization of the developer experience presents a significant commercial opportunity—good toolchains can lower the barrier to Agent development

from "needing to understand the entire protocol stack" to "dragging and dropping a few components." Official tools lay the "foundation," and the market builds the "house" on top.

15.5 Vertical-Specific Agents

Analogy: SaaS products for various industries

Specialized Agents for specific scenarios.

Anticipated verticals:

Vertical	Agent Type	Core Capabilities
Quantitative Trading	Arbitrage Agent, Market Maker Agent	Cross-branch spread discovery, high-frequency trade execution
Customer Service	FAQ Agent, Ticket Agent	RAG Q&A, automatic ticket classification and routing
Marketing & Promotion	Content Agent, Ad Placement Agent	Automated content generation, optimization of ad placement strategies
Legal Consulting	Compliance Agent, Contract Agent	Contract clause review, compliance checks
Code Development	Coding Agent, Review Agent	Automated coding, code review, test generation
Data Analysis	Analysis Agent, Report Agent	Data cleaning, statistical analysis, visual report generation

Dependent primitives:

- DID identity (Agent and client identities)
- ACTUS contracts (service contracts and payment settlement)
- Compute trading protocol (invoking LLMs to perform actual work)

Emergence logic: This is the most direct commercial scenario. Each vertical-specific Agent is an "autonomous employee that can earn money." Developers build Agents, deploy them on-chain, and Agents generate revenue by providing services. Developers don't need to do marketing—the task marketplace automatically matches supply and demand.

15.6 "Bus" Service: Hosted Strategy Pool

Analogy: Decentralized fund / robo-advisor

Asset custody and strategy execution services for retail users.

How it works:

- Professional developers build high-performance trading Agents.
- Retail users delegate funds to an Agent (rights and obligations defined via ACTUS contract).
- The Agent executes trading strategies on behalf of the user.
- Profits are shared according to the contract agreement.

Dependent primitives:

- ACTUS contracts (delegated management contracts defining profit-sharing ratios, redemption conditions, risk control boundaries)
- DID identity (reputation score of the strategy provider)
- Judicial primitives (handling breach disputes)

- Compliance slots (in some jurisdictions, this may be considered a fund product requiring compliance checks)

Emergence logic: Most retail users lack the capability to build and operate Agents, but they have capital. The "bus" service allows retail users to "ride along"—paying a fare (management fee) to benefit from the returns of professional Agent strategies. This aligns with the logic of the traditional fund industry but with execution and transparency far exceeding traditional funds.

Risk Note: Such services may be classified as Collective Investment Schemes in many jurisdictions, requiring corresponding licenses. The design of compliance slots allows branch operators to perform compliance checks as required by local regulations.

15.7 Arbitration Services

Analogy: Decentralized commercial arbitration institutions

Third-party arbitration services handling everyday commercial disputes.

Relationship with system-level judicial primitives:

- System-level judicial primitives (AI Traffic Police + AI Court) handle **protocol-level violations** (e.g., insufficient margin, SLA breaches).
- Arbitration services handle **commercial-level disputes** (e.g., "task quality not meeting standards," "deliverable not matching description").

How it works:

- Disputing parties agree to submit to arbitration (arbitration clause pre-defined in the ACTUS contract).

- The arbitration Agent reviews evidence and hears statements from both sides.
- A ruling is made based on contract terms and on-chain evidence.
- The ruling is automatically enforced via the ACTUS contract.

Dependent primitives:

- ACTUS contracts (arbitration clauses and ruling enforcement)
- DID identity (qualifications and reputation of arbitrators)
- Economic primitives (settlement of arbitration fees)

Emergence logic: Commercial disputes are a constant in any market economy. System-level judicial primitives handle "major issues" (protocol-level violations) and are not suited for everyday commercial disputes. Third-party arbitration services fill this gap, enabling quick, low-cost resolution of commercial disputes between Agents.

15.8 Identity Verification Services

Analogy: Decentralized identity verification platform

Identity verification and human-Agent differentiation services built on top of DID.

Anticipated service types:

- **KYC Services:** Provide identity verification for branches requiring compliance (linking DID to real-world identity).
- **Human-Agent Verification:** Prove whether a DID is backed by a human or an Agent (some scenarios require human participation).
- **Qualification Certification:** Prove that an Agent possesses specific qualifications (e.g., "a trading Agent that has undergone security audit").

- **Reputation Aggregation:** Cross-platform reputation integration (mapping off-chain reputation to on-chain DID).

Dependent primitives:

- DID identity (foundation for all identity verification)
- Compliance slots (interface with compliance requirements of various jurisdictions)

Emergence logic: Identity is the cornerstone of trust. As on-chain transaction volume grows, "Is this Agent trustworthy?" will become a common need. Identity verification services address this need with standardized solutions by attaching Verifiable Credentials to DIDs.

Summary

The eight anticipated emergent markets above cover the key elements of a sound economy:

Element	Emergent Market
Labor Market	Task Marketplace
Risk Management	Insurance Contracts
Credit System	Credit Investigation & Rating
Infrastructure	AI Toolchain
Industrial Specialization	Vertical-Specific Agents
Asset Management	“Bus” Service
Dispute Resolution	Arbitration Services
Trust Foundation	Identity Verification Services

These markets do not need to be built by the DioChain team. They will be spontaneously discovered and filled by market participants when the primitive set is sufficiently complete and the user base is large enough. Our job is to ensure the primitive set lacks no necessary components—if any of the above markets cannot be built on the existing primitive set, that is a design flaw requiring the addition of new primitives.

{ “translated”: "# Chapter 16: Risk Analysis\n\n> Transparent risk disclosure is a fundamental prerequisite for the credibility of a white paper.\n\n—\n\nThis chapter systematically lists the main risks faced by DioChain, along with corresponding mitigation strategies. Some risks can be eliminated through architectural design, some can only have their probability reduced, and others we acknowledge cannot be fully resolved.\n\n—\n\n## 16.1 Technical Risks\n\n### 16.1.1 Non-Determinism of AI Consensus\n\n**Risk Description:** DioChain’s Evolution Engine (Control Plane) relies on LLMs for rule update proposals and judicial rulings. However, LLM outputs are non-deterministic—the same prompt may yield different results from LLMs on different nodes. This fundamentally conflicts with the deterministic execution of traditional blockchains (same input = same output).\n\n**Severity: High**\n\n**Mitigation Strategies:**\n- **Structured Output Constraints:** The AI Control Plane’s outputs use a structured parameter adjustment proposal format (e.g., "adjust the margin threshold from 10% to 12%"), not free text. Structured outputs significantly reduce the space for non-determinism.\n- **Semantic Majority Consensus:** Multiple nodes independently run AI inference, reaching consensus on outputs at a semantic level (not bit-level). For example, if 3/5 of nodes propose the direction "increase margin," even if the specific values differ slightly, a consensus can be reached by taking the median value.\n- **TEE Remote Attestation:** Critical AI inference runs within a Trusted Execution Environment (TEE). TEE remote attestation ensures the inference process has not been tampered with.\n- **Hard-coded Bounds:** AI parameter adjustment ranges have hard-coded upper and lower limits. Even if an AI proposes extreme suggestions, they will be intercepted by these hard-coded bounds.\n\n**Residual Risk:** Semantic majority consensus is a relatively new concept, and its reliability in large-scale deployments has not been fully

validated.\n\n### 16.1.2 Liquidity Fragmentation Among Branches\n\n**Risk Description:** The L1 layer consists of multiple independent branches, each maintaining its own liquidity. If liquidity is dispersed across many small branches, large transactions may not be completed within a single branch, increasing cross-branch settlement delays.\n\n**Severity: Medium**\n\n**Mitigation Strategies:**\n- **AI-Driven Liquidity Routing:** AI automatically splits large transactions into multiple smaller ones, routing and executing them across multiple branches.\n- **HTLC Atomic Swaps:** Use Hash Time-Locked Contracts (HTLCs) to ensure atomicity of cross-branch transactions—either all complete or all roll back.\n- **Natural Aggregation:** Market forces naturally drive liquidity to aggregate towards high-quality branches. Poorly operated small branches will exit due to lack of traffic.\n\n**Residual Risk:** In the early stages of the network (with few branches and low liquidity), large transactions may face significant slippage.\n\n### 16.1.3 Performance Ceiling of WASM VM\n\n**Risk Description:** DioChain’s smart contracts run on the WASM virtual machine. While WASM performs better than EVM, it still incurs virtualization overhead. Complex calculations for ACTUS contracts (e.g., Monte Carlo simulations, Value at Risk calculations) may execute too slowly within the VM.\n\n**Severity: Medium**\n\n**Mitigation Strategies:**\n- **ACTUS Precompiled Contracts:** Implement the most commonly used ACTUS contract types (PAM, ANN, NAM, etc.) as precompiled contracts at the L0 layer, executing directly at the native layer to bypass VM overhead.\n- **Off-Chain Computation + On-Chain Verification:** Complex calculations are performed off-chain, with only results and proofs submitted for on-chain verification.\n- **Progressive Optimization:** Initially support only ACTUS contract types with lower computational complexity (PAM, Swap), gradually supporting more complex types as performance is optimized.\n\n**Residual Risk:** Precompiled contracts are hard-coded; adding new ACTUS types requires a protocol upgrade (hard fork or on-chain governance vote).\n\n—\n\n## 16.2 Economic Risks\n\n### 16.2.1 Death Spiral (Procyclical Effect)\n\n**Risk Description:**

DioChain’s Evolution Engine can dynamically adjust economic parameters (e.g., margin thresholds, fee rates, liquidity incentives). However, during sharp market downturns, AI’s dynamic parameter adjustments may **accelerate** a crash rather than prevent it:\n\n- Token price falls → AI raises margin threshold (for safety) → branches are forced to sell tokens to supplement margin → token price falls further → ...\n\nThis is the classic procyclical effect, which has occurred multiple times in history:\n- **Luna/UST (2022):** Death spiral of an algorithmic stablecoin, $40B evaporated.\n- **1987 Black Monday:** Programmed trading accelerated the stock market crash.\n- **2008 Financial Crisis:** Margin calls triggered chain reactions of selling.\n\n**Severity: High**\n\n**Mitigation Strategies:**\n- **Hard-coded Bounds:** AI parameter adjustment ranges have hard-coded upper and lower limits. For example, the single adjustment range for the margin threshold cannot exceed ±2%, regardless of what the AI thinks it should be.\n- **Circuit Breaker Mechanism:** When market volatility exceeds a threshold, pause AI parameter adjustments and enter "manual mode" where decisions are made by a governance committee.\n- **Sandbox Validation:** AI parameter adjustment proposals undergo stress testing in a sandbox environment (including extreme market scenarios) and only take effect on the mainnet after passing.\n- **Anti-Procyclical Design:** Build anti-procyclical preferences into the AI’s objective function—during downturns, it tends to relax conditions rather than tighten them.\n\n**Residual Risk:** Hard-coded bounds may limit the system’s self-protection ability in extreme situations. The circuit breaker’s trigger threshold may also be set inappropriately—too sensitive causing frequent triggers, too sluggish rendering it ineffective.\n\n### 16.2.2 Nested Margin Risk\n\n**Risk Description:** To improve capital efficiency, DioChain allows the use of productive assets (e.g., Liquid Staking Tokens / LST) as margin. However, if the underlying asset de-pegs (e.g., stETH de-pegging from ETH), the actual value of the margin may shrink sharply, leading to systemic insuffic-

iency.\n\n**Severity: Medium-High**\n\n**Mitigation Strategies:**\n- **Margin Asset Whitelist:** Only assets approved by governance can be used as margin; new assets must pass strict risk assessments.\n- **Diversification Requirement:** A single asset cannot exceed 50% of a branch’s total margin, enforcing diversification.\n- **Haircut (Discount Rate):** Productive assets are counted towards margin value at a discount below face value. For example, 1 stETH is only counted as 0.9 ETH of margin value.\n- **Real-Time Revaluation:** Margin value is revalued in real-time at market price; if it falls below the threshold, a margin call notification is automatically triggered.\n\n**Residual Risk:** Under extreme market conditions (e.g., systemic DeFi risk events), all productive assets may de-peg simultaneously, rendering diversification ineffective.\n\n### 16.2.3 Capital Efficiency Dilemma\n\n**Risk Description:** The margin system requires branches to lock up a large amount of tokens, which become "dead money"—unavailable for other productive uses. This reduces the overall capital efficiency of the system and may deter potential branch operators.\n\n**Severity: Medium**\n\n**Mitigation Strategies:**\n- **Productive Margin:** As mentioned, allow the use of productive assets like LST as margin, enabling the margin to generate yield even while locked.\n- **AI Dynamic Adjustment of Margin Thresholds:** The Evolution Engine dynamically adjusts margin requirements based on market conditions—lowering thresholds to release liquidity during stable markets, raising thresholds to increase safety cushions during volatility.\n- **Tiered Margin:** Distinguish between core margin (must be highly liquid assets) and supplementary margin (can be various productive assets), with lower requirements for core margin.\n\n**Residual Risk:** Dynamically adjusting margin thresholds may introduce new procyclical risks (see 16.2.1) and needs to be used in conjunction with circuit breaker mechanisms.\n\n—\n\n## 16.3 Security Risks\n\n### 16.3.1 Data Poisoning\n\n**Risk Description:** The Evolution Engine relies on on-chain data (trading volume, price, default rates, etc.) to formulate rule upd-

ate proposals. Attackers can "poison" the AI’s input data by creating a large number of fake transactions (wash trading) or abnormal patterns, inducing the AI to make incorrect rule updates.\n\n**Severity: High**\n\n**Mitigation Strategies:**\n- **Sandbox Backtesting:** All AI rule update proposals are backtested on historical data; if they perform poorly in past extreme markets, the proposal is rejected.\n- **Staking Game:** Rule update proposals require the proposer to stake margin. If the proposal causes system losses after implementation, the proposer’s margin is slashed. This imposes an economic cost for submitting malicious proposals.\n- **Hard-coded Bounds:** Even if the AI is induced, its output cannot exceed the hard-coded parameter ranges.\n- **Anomaly Detection:** Perform statistical anomaly detection on on-chain data to filter out obvious wash trading and market manipulation.\n\n**Residual Risk:** Carefully designed data poisoning may evade simple anomaly detection. Adversarial AI is an active research area with no universal defense solution.\n\n### 16.3.2 Adversarial Attacks\n\n**Risk Description:** Attackers may create fake transaction graphs to deceive the AI Traffic Police. For example: constructing a series of transactions that appear normal but are actually money laundering paths, tricking the AI Traffic Police into judging them as legitimate.\n\n**Severity: High**\n\n**Mitigation Strategies:**\n- **Multi-Node Independent Inference:** Multiple AI Traffic Police nodes independently analyze the same transaction; the consensus mechanism requires majority agreement to approve.\n- **Adversarial Training:** Regularly train the AI Traffic Police with known attack patterns to improve its detection capabilities.\n- **Human Jury Final Confirmation:** For high-value or high-risk judgments, introduce a human jury for final confirmation.\n- **Continuous Monitoring and Tracing:** Even if a transaction passes the AI Traffic Police’s real-time check, subsequent batch analysis may still identify issues and trigger retrospective handling.\n\n**Residual Risk:** There is an eternal arms race between attackers and defenders. We cannot guarantee the AI Traffic Police will never be circ-

umvented. Human juries are the final safety net but introduce trade-offs in efficiency and scalability.\n\n### 16.3.3 Hardware Centralization (GPU Monopoly)\n\n**Risk Description:** "Decentralized" AI chains still rely on high-performance GPUs at the underlying layer, and the GPU market has a highly concentrated supplier landscape. If major hardware suppliers change licensing terms, restrict supply, or face export controls, the foundation of the entire decentralized AI ecosystem could be shaken.\n\n**Severity: Medium**\n\n**Mitigation Strategies:**\n- **Multi-Hardware Support:** Support multiple inference hardware types at the protocol level (GPUs from various manufacturers, dedicated accelerator chips, consumer-grade SoCs, custom ASICs, etc.) to avoid dependency on a single supplier.\n- **Encourage Edge Computing:** Support consumer-grade hardware participation in the network (e.g., using consumer-grade SoCs to run small models), lowering the entry barrier.\n- **Model Quantization Support:** Support quantized models (INT4/INT8), enabling mid-to-low-end hardware to participate in inference.\n\n**Residual Risk:** In the short term, high-performance AI inference remains heavily dependent on a few hardware suppliers. This is a structural risk faced by the entire industry, which a single project cannot solve independently.\n\n—\n\n## 16.4 Governance Risks\n\n### 16.4.1 AI Court Misjudgment\n\n**Risk Description:** The AI judicial system (AI Traffic Police + AI Court) may make erroneous rulings due to model hallucination or training data bias, leading to legitimate users' assets being incorrectly frozen or slashed.\n\n**Severity: High**\n\n**Mitigation Strategies:**\n- **Human Jury Final Confirmation Right:** All major rulings (involving large asset freezes or slashing) must be confirmed by a human jury.\n- **Time-Limited Freezes:** Asset freezes have a maximum time limit (e.g., 72 hours); if not confirmed within that time, they are automatically unfrozen.\n- **Multi-Level Appeal Mechanism:** Defendants have the right to appeal; appeals are reviewed by a different AI node + human jury panel.\n- **Misjudgment Compensation:** If ultimately ruled a misjudgment, the defendant can receive compensation from

the system’s insurance fund.\n- **Case Law Database:** All rulings are transparent and public, forming a case law database for reference in similar future cases.\n\n**Residual Risk:** Human jury involvement introduces subjectivity and efficiency issues. In extreme cases (e.g., massive attacks causing a flood of cases), human juries may become a bottleneck.\n\n### 16.4.2 Politicization of Compliance Slots\n\n**Risk Description:** The design intent of Compliance Slots is to allow branches to adapt to local regulations. In practice, however, some jurisdictions may use Compliance Slots for political censorship—for example, requiring all branches to block users from specific regions or prohibit certain types of transactions.\n\n**Severity: Medium**\n\n**Mitigation Strategies:**\n- **Branch-Level Enforcement:** Compliance Slots only take effect at the branch level, not the protocol level. One branch’s compliance rules do not affect other branches.\n- **User Freedom of Choice:** Users can choose unrestricted branches (if their jurisdiction allows). Compliance is a branch’s obligation, not a user’s shackle.\n- **Transparency Requirement:** Branches must publicly disclose their Compliance Slot configurations; users can understand a branch’s compliance policies before choosing it.\n\n**Residual Risk:** This is a political issue that architecture cannot fully resolve. If a jurisdiction mandates that **all** branches operating within its territory enforce censorship, the user’s only choice is to use offshore branches—but these may have higher latency and worse service. We acknowledge: technological neutrality does not equal political neutrality. The protocol does not censor, but the protocol cannot prevent branches from censoring.\n\n—\n\n## 16.5 Market and Competition Risks\n\n### 16.5.1 Impact of Cost Advantage from Centralized Inference Services\n\n**Risk Description:** Low-cost inference services provide capabilities close to frontier commercial models at extremely low costs, triggering collective reflection in the AI+Crypto space: if centralized AI services are already cheap and fast enough, what is the point of the blockchain layer? This trend has led to a significant overall decline in AI+Crypto tokens.\n\n**Severity: High**\n\n**Response:**\n\nThe answer to this question dep-

ends on "what the blockchain layer provides."\n\nIf the blockchain layer only provides "decentralized compute" (like Bittensor, Render Network), then the low-cost inference impact indeed poses a major threat—centralized compute is cheaper, faster, and more stable.\n\nBut DioChain's blockchain layer provides **social infrastructure**:\n\n| Need | Centralized AI's Answer | DioChain's Answer |\n|------|------------------------|------------------|\n| Identity | API Key (revocable, traceable) | DID (self-sovereign identity) |\n| Contract | User Agreement (platform-dominated) | ACTUS contract (on-chain executable, bilateral equality) |\n| Judiciary | Platform Customer Service (uncertain response) | AI Judicial System (rule-based, appealable) |\n| Payment | Credit Card (excludes unbanked populations) | Token (global, no barriers) |\n| Compliance | Platform Uniform Policy (one-size-fits-all) | Compliance Slots (localized adaptation) |\n\nLow-cost inference services can make AI inference cheaper, but they cannot give an Agent an irrevocable identity, provide cross-border instant settlement, or enable two mutually distrusting Agents to sign an enforceable contract. These are the irreplaceable values of the blockchain layer.\n\n### 16.5.2 First-Mover Advantage of Existing Competitors\n\n**Risk Description:** Multiple competitors have established significant first-mover advantages in their respective fields.\n\n| Competitor | First-Mover Advantage | DioChain's Differentiation |\n|---|---|---|\n| **Bittensor (TAO)** | 129+ subnets, Grayscale trust backing | Bittensor's consensus is actually centralized (Opentensor Foundation's PoA), subnet quality is hard to verify. DioChain does not compete on compute; it builds financial infrastructure. |\n| **Ritual** | TEE+ZKP for AI inference verification | Still in pre-mainnet stage, architecture not validated at scale. DioChain does not reinvent the wheel for AI inference verification; it can integrate similar technologies from compute markets. |\n| **NEAR AI** | 1M TPS, NEAR Intents processing $6B+ | AI strategy is more narrative-driven market positioning; technical architecture is not natively designed for AI Agents. |\n| **Fetch.ai / ASI Alliance** | Achieved AI-to-AI payment | Internal alliance fragmentation,

repeated pivots, lack of clear long-term direction. |\n| **ACTUS Protocol (ETH)** | Has Solidity-implemented ACTUS | Just a contract library, no chain-level optimization, no supporting identity/judiciary/compliance infrastructure. |\n| **Casper Network** | Lists ACTUS as a priority | Low market cap, low adoption, ACTUS integration still in planning stage. |\n| **EZKL** | Most mature zkML framework, used by multiple enterprises | Focuses on ZK proofs, not a direct competitor, potential technical partner. |\n\n**DioChain's Positioning Difference:** Most of the above competitors focus on a single dimension (compute market, AI inference verification, contract standard). DioChain's positioning is **integrated Agent social infrastructure**—providing a complete, self-consistent system. The risk of this positioning is "trying to do everything = doing nothing well," but we mitigate this risk by "defining primitives, not applications"—only building the minimal complete set of primitives and letting the market build applications on top.\n\n### 16.5.3 Overall Credibility Crisis in the AI+Crypto Space\n\n**Risk Description:** The AI+Crypto space has experienced a severe credibility crisis: many projects have narratives but no products, tokens have significantly corrected, and investor confidence is severely damaged. New projects face widespread trust deficits.\n\n**Severity: Medium-High**\n\n**Response:**\n\nOur attitude towards this is: **focus on infrastructure delivery,

Chapter 17: Roadmap

> This roadmap is triggered by milestone conditions, not date commitments.

Principles

This project does not set date commitments such as "Mainnet launch in Q3 2025." Software development is not an assembly line; delivery timelines for complex systems cannot be precisely predicted.

This roadmap is triggered by **milestone conditions**: when conditions are met, progress to the next phase. If conditions are not met, do not force advancement.

Phase 0: Proof of Concept

Status: In Progress

Deliverables:

- White paper release (this document)
- Formal specification of the core primitive set
- PoC prototype: Demonstrates the complete process of Agent registering DID → signing ACTUS contract → invoking compute → settlement in a local environment
- Economic model simulation: Uses Monte Carlo simulation to verify the robustness of Token economics under extreme scenarios

Conditions to advance to the next phase:

- [ ] PoC prototype passes internal review
- [ ] Economic model simulation shows no death spiral in 1000 extreme scenarios
- [ ] Obtain preliminary architecture review from at least 1 external security team

---

Phase 1: Testnet

Deliverables:

- Public testnet launch
- L0 settlement layer + 3-5 official L1 branches
- Core ACTUS contract templates: PAM (Principal at Maturity) + Swap
- DID registration and reputation system v1
- AI Traffic Police v1 (basic rule engine + LLM assistance)
- Compute trading protocol v1 (supplier registration + simple routing + streaming payment)
- Open-source Agent SDK + 3-5 standard Agent templates
- diolaw v0.1 (off-chain voting accumulation prototype + basic legislative process)
- dioproof v0.1 (ZK-TLS evidence packaging PoC, integrated with AI Traffic Police)

Conditions to advance to the next phase:

- [ ] Testnet runs stably for > 90 days with no L0 layer consensus failures
- [ ] Daily average compute invocation volume > 100M tokens
- [ ] Active Agents > 1,000
- [ ] Complete at least 100 full lifecycles of ACTUS contracts
- [ ] Pass contract audit by at least 1 professional security audit firm
- [ ] AI Traffic Police misjudgment rate < 5% (based on human-annotated test set)

Phase 2: Mainnet Launch

Deliverables:

- Mainnet launch, Token officially issued
- Compute market officially opens (third-party suppliers can register)
- Open L1 branch registration (third-party entities can stake margin to open branches)
- ACTUS contract type extensions: ANN (Annuity), NAM (Negative Amortization), LAM (Linear Amortization)
- AI Court v1 (supports dispute arbitration and appeals)
- Compliance Slots v1 (supports basic KYC/AML plugins)
- Cross-branch liquidity routing optimization
- diolaw officially launched (complete loop of off-chain lobbying + on-chain referendum)
- dioproof officially launched (supports evidence types such as bank transfers, chat records, webpage snapshots)

Conditions to advance to the next phase:

- [ ] Mainnet runs stably for > 180 days
- [ ] Third-party branches > 10

- [ ] Third-party compute suppliers > 20
- [ ] Daily average compute invocation volume > 1B tokens
- [ ] Active Agents > 10,000
- [ ] At least 3 community-built emergent markets appear (any 3 listed in Chapter 15)

Phase 3: Ecosystem Maturity

Deliverables:

- Advanced ACTUS contract types: OPTNS (Options), FUTUR (Futures), CEG (Credit Enhancement Guarantee), etc.
- Cross-chain bridging: Asset interoperability with mainstream L1s
- AI Evolution Engine fully enabled: Complete governance loop of AI dynamic parameter adjustment + community governance + human jury
- Compliance Slots v2: Supports compliance frameworks of major jurisdictions
- Decentralized governance: Foundation gradually exits core decision-making, power transferred to the community
- Sovereign Chain Fork Kit released
- Cross-chain Agent pass-through protocol v1
- diolaw sovereign chain adaptation (PoP voting mode)

Continuous iteration:

- Continuously expand ACTUS contract types based on community needs
- Continuously upgrade AI Evolution Engine based on AI technology progress
- Continuously update Compliance Slots based on regulatory environment changes
- Gradually scale down the foundation’s official branches when they are no longer the main source of traffic

---

Non-Goals

The following items are not in the roadmap because they are not what DioChain should do:

- **Building specific applications**: We only build primitives, not applications. Applications emerge from the market.
- **Issuing stablecoins**: Token anchoring strategies are determined by governance, but DioChain does not issue stablecoins itself.
- **Operating exchanges**: Branches provide trading functions, but DioChain does not operate centralized exchanges.
- **Providing compute**: The compute trading center is a trading protocol, not a compute cloud.
- **Creating Token scarcity**: $DIO is an elastic supply unit of account, not a scarce collectible. DioChain does not artificially create scarcity to drive up prices.

Chapter 18: Conclusion

Positioning Statement

The blockchain industry is not short of speculative tools. What it lacks is infrastructure—the infrastructure that enables AI Agents to have identity, sign contracts, resolve disputes, acquire computing power, and participate in economic activities.

The core mission of DioChain can be stated in one sentence: **To build financial infrastructure for the Agent era**.

The goal of DioChain is not "to use AI for trading to make more money," nor is it "to decentralize everything" or "to disrupt traditional finance."

Its core value lies in providing the most fundamental social structures necessary for the operation of an emerging Agent economy: identity, currency, contracts, law, and markets.

Furthermore, DioChain is not a single chain, but a set of social operating system protocols. The same set of protocols can be forked into parallel universes—Utopia Chain serves as the ideal-state reference implementation, while various Sovereign Chains operate independently under different governance configurations. The protocol is unified; governance is diverse.

Minimal Complete Set of Primitives

Our design philosophy is **to define primitives, not applications**.

Six groups of primitives—Economic Primitives, Contract Primitives (ACTUS), Identity Primitives (DID), Judicial Primitives, Compliance Slots, and the Compute Trading Protocol—form a minimal complete set. "Minimal" means we do not do anything extra; "complete" means that on top of these primitives, the market can spontaneously give rise to any applications it needs.

The set of primitives defines the rules, but rules require infrastructure to carry and execute them. aibank provides access tools, agenter acts as a super-scheduler orchestrating Agent behavior, diolaw carries the legislation and referendum process, dioproof provides zero-knowledge proof infrastructure—these four form a closed loop, enabling the set of primitives to operate, the rules to evolve, and disputes to be adjudicated.

The same set of primitives can operate on multiple parallel chains under different governance configurations. The primitives remain unchanged; the governance parameters are variable—this is the core hypothesis of the multi-chain topology.

We do not know what the most successful Agent application of the future will look like. But we believe that, whatever it looks like, it will need identity, contracts, payments, justice, and computing power.

We build these. The rest, we leave to the market.

Economic Sustainability

The economic model of DioChain does not rely on Token price appreciation to sustain its operation.

The elastic supply mechanism of $DIO ensures that the Token supply scales in sync with the size of the economy—there is no artificial scarcity, no inflation schedule, and every Token minted has a real counterparty consideration. This means the system’s value creation stems from economic activity itself, not from speculative expectations.

Revenue sources are clear and sustainable:

- **Branch Fees**: Direct income for branch operators, priced by market competition.
- **Compute Market Transaction Volume**: Every Agent inference call is a real economic activity, generating real settlement demand.
- **Social Treasury**: Through the diolaw legislative mechanism, the community can levy transaction taxes on commercial activities, providing a sustainable funding source for public services (judicial system, infrastructure maintenance).
- **Official Infrastructure Services**: Tools like aibank and agenter are charged on-demand or provided for free, with pricing authority belonging to the operator.

Investors should focus on core metrics not Token price, but rather: Total Value Locked (TVL), Daily Active Agents (DAA), daily tokens processed (daily compute consumption), and governance activity through diolaw legislation. These metrics directly reflect the real prosperity of the Agent society.

Current Stage

DioChain is currently in the Phase 0 (Proof of Concept) stage. The whitepaper is the starting point, not the end point.