



# 接口文档

## 1. 区块链底层组 (Blockchain Core)

### 成员 1: 区块链核心开发 (数据层)

**核心职责:** 定义数据结构, 维护链的完整性。

阶段	接口名	输入	输出	处理逻辑
数据定义	<code>Transaction</code> , <code>Block</code> , <code>Account</code> (类定义)	无	<code>Transaction</code> , <code>Block</code> , <code>Account</code> 类	定义 Pydantic 模型或 Python 类。
哈希计算	<code>calculate_hash</code> , <code>get_merkle_root</code>	任意字符串/字节流	哈希值 (String)	实现 SHA256 哈希算法; 实现 Merkle Tree 根哈希计算。
链维护	<code>Blockchain.add_block</code>	<code>Block</code> 对象 (来自成员 2 的挖矿结果)	更新后的区块链列表	验证区块哈希链接 ( <code>prev_hash</code> ); 将合法区块追加到 <code>self.chain</code> 列表。

### 成员 2: 区块链核心开发 (执行层)

**核心职责:** 交易执行与状态更新。

阶段	接口名	输入	输出	处理逻辑
交易池管理	<code>Blockchain.add_transaction</code>	<code>Transaction</code> 对象	待打包交易池	验证交易签名 ( <code>tx.verify()</code> ); 检查 Nonce 防止重放; 加入 <code>pending_transactions</code> 。
挖矿/出块	<code>Blockchain.mine_block</code>	<code>pending_transactions</code> (内部状态)	新生成的 <code>Block</code> 对象	从池中取出交易; 调用 <code>StateProcessor</code> 执行交易; 打包生成新 <code>Block</code> 。
状态更新	<code>StateProcessor.apply_transaction</code>	<code>Transaction</code> 对象	更新后的 <code>WorldState</code>	解析交易 Payload; 调用成员 3/4 的合约接口更新全局状态 ( <code>WorldState</code> )。

## 2. 智能合约组 (Smart Contracts)

### 成员 3: 智能合约开发 (流程控制)

**核心职责:** 运维 SOP 状态机逻辑。

阶段	接口名	输入	输出	处理逻辑
SOP 状态流转	<code>OpsContract.submit_analysis</code>	<code>Transaction</code> (类型: <code>SUBMIT_ANALYSIS</code> , <code>PROPOSE_ROOT_CAUSE</code> )	更新后的 SOP 状态 ( <code>Data_Collected</code> 等)	检查当前 SOP 阶段 ( <code>current_state</code> ) 是否允许该操作; 更新状态机。
事件记录	<code>Event</code> (数据结构)	业务操作结果	<code>Event</code> 对象 (供前端查询)	生成结构化日志, 记录关键步骤 (如“分析已提交”)。

## 成员 4: 智能合约开发 (经济模型)

核心职责: Token 管理与共识计算。

阶段	接口名	输入	输出	处理逻辑
Token 管理	<code>TokenContract.transfer</code> , <code>TokenContract.stake</code> , <code>TokenContract.slash</code>	<code>Transaction</code> (类型: <code>TRANSFER</code> , <code>STAKE</code> )	更新后的 <code>balances</code> 字典	检查余额; 执行转账/质押/扣除操作; 更新账本。
共识投票	<code>GovernanceContract.vote</code>	<code>Transaction</code> (类型: <code>VOTE</code> )	提案状态 ( <code>Passed / Rejected</code> )	记录投票; 计算权重 ( <code>质押量 * 信誉分</code> ); 判断是否达到阈值。

### 3. Agent 适配组 (Agent Adapter)

#### 成员 5: Agent 中间件开发

**核心职责:** 链上交互中间件。

阶段	接口名	输入	输出	处理逻辑
交易封装	<code>Transaction.sign</code> (调用)	Agent 的决策 ( <code>Action</code> , <code>Payload</code> )	签名后的 <code>Transaction</code> 对象	构造 <code>Transaction</code> 对象; 调用 <code>sign()</code> 签名。
上链交互	<code>ChainClient.send_transaction</code>	<code>Transaction</code> 对象	交易回执 ( <code>Receipt</code> )	调用成员 2 的 <code>add_transaction</code> 接口发送交易; 轮询等待回执。

#### 成员 6: Agent 适配与 Prompt 工程

**核心职责:** Agent 身份与 Prompt 改造。

阶段	接口名	输入	输出	处理逻辑
身份管理	<code>AgentProfile</code> (数据结构)	无	<code>AgentProfile</code> (包含地址、私钥)	生成/加载钱包私钥; 为每个 Agent 实例分配身份。
Prompt 改造	<code>SystemPrompt</code>	原始运维任务描述	结构化的 Agent 输出 (包含 <code>action_type</code> , <code>stake_amount</code> )	修改 System Prompt, 注入“质押”、“经济激励”等约束; 解析 LLM 输出为 JSON。

## 4. 前端交互组 (Frontend & API)

### 成员 7: 前端开发 (区块链浏览器)

**核心职责:** 区块链数据可视化。

阶段	接口名	输入	输出	处理逻辑
后端 API 开发	GET /api/blocks , GET /api/transaction/{hash}	Blockchain 对象 (来自成员 1/2)	JSON 数据	封装 FastAPI 接口, 暴露区块和交易数据。
前端展示	BlockExplorer 组件	API 返回的 JSON 数据	渲染区块列表、交易详情页、Merkle 树验证视图	调用 API 获取数据并渲染。

### 成员 8: 前端开发 (运维控制台)

**核心职责:** 业务状态与经济看板。

阶段	接口名	输入	输出	处理逻辑
后端 API 开发	GET /api/stat e/sop , GET /a pi/state/agent s	OpsContract , TokenContract 状态 (来自成员 3/ 4)	JSON 数据	封装 FastAPI 接 口, 暴露 SOP 状 态和 Agent 资产信 息。
前端展示	SOPDashboard , EconomyMonito r 组件	API 返回的 JSON 数据	渲染 SOP 流程图 (React Flow)、资 产图表 (ECharts)	调用 API 获取数据 并渲染。