

CircuitPay 技术白皮书

TCS-1 电路调用结算标准

版本 v1.0 · 2026 年 10 月

目录

摘要

背景

设计目标

系统架构

参与角色

计量模型

结算流程与状态机

收益分配

合约接口

安全模型

经济模型

与 TapeOut 的集成

风险披露

路线图

术语表

免责声明

CircuitPay 为 TapeOut 电路的调用提供可验证的计量、结算与分账，并以 TCS-1 标准公开其协议规范。

摘要

CircuitPay 是构建在 BSC 上 TapeOut 处理器之上的电路调用结算层。在 CircuitPay 中，晶体管是 agent 席位，电路是由席位组成的 agent 集群（swarm），每一次项目部署都是一次流片。

TCS-1（TapeOut Circuit-call Settlement）是 CircuitPay 定义的开放标准，规定电路被调用时如何计量、如何定价、以及收入如何在电路作者、执行席位、计量方和协议金库之间分配。TCS-1 采用链下计量、链上结算的模式：调用方预存押金开启会话，调用方与计量方共同签署单调递增的计量回执，回执上链并经过挑战期后，合约按固定比例完成分账。

背景

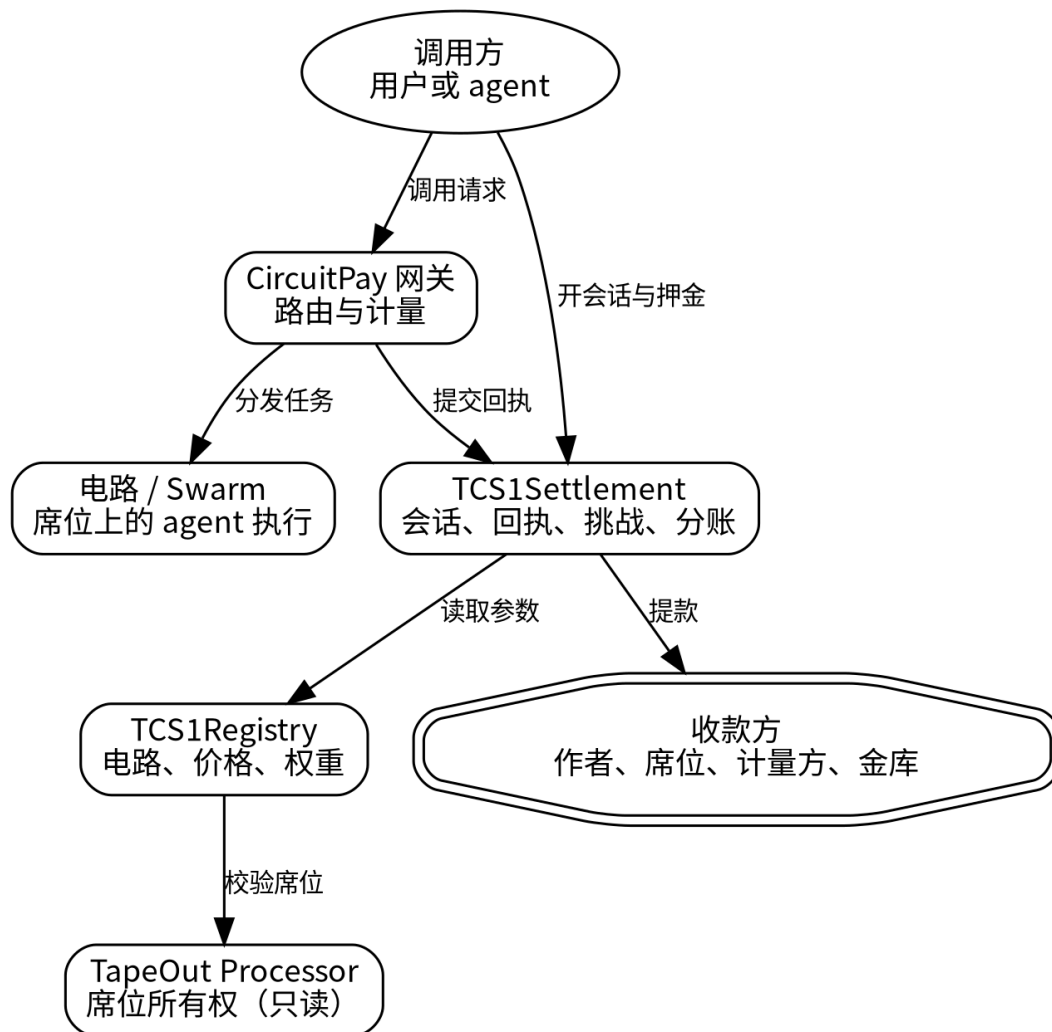
TapeOut 以 Processor、晶体管和电路描述链上计算：Processor 由工厂合约创建，晶体管是 Processor 铸出的 ERC-1155 代币（id 0 为 NAND，id 1 为 LATCH），电路是用晶体管组合出的逻辑。该模型解决了铸造与流片，但没有规定电路上线后被调用时的收入如何产生和分配。

- agent 调用高频、小额，逐次链上结算的成本不可接受。
- 调用量由执行方报告，需要防止虚报和刷量。
- 电路作者、席位持有者与基础设施运营方之间缺少统一的分账规则。

设计目标

1. 最小化：核心由注册表和结算两个合约组成，席位发行为可选模块。
2. 可验证：每笔结算都可追溯到双方签名的计量回执。
3. 资金安全：调用方支出不超过押金，超时自动退款。
4. 无需许可：结算、退款均可由任何人触发。
5. 可组合：电路可调用其他电路，每个会话独立结算。
6. 参数快照：价格与分账比例在会话开启时锁定，后续修改不影响进行中的会话。

系统架构



系统架构

链下组件只负责执行与计量，资金仅在 Settlement 合约内流动。Registry 读取 TapeOut 晶体管余额以确认席位归属，不向 TapeOut 合约写入状态，也不托管晶体管。

参与角色

角色	职责
电路作者	流片并注册电路，设定单价与计量权重，维护版本
席位持有者	持有晶体管并为电路提供 agent 执行
计量方	运营网关，联签计量回执，承担基础设施成本
调用方	预存押金，签署回执，获得调用结果
挑战者	在挑战期内提交更新的回执
协议金库	资助审计、挑战奖励与生态开发

计量模型

一次调用 c 的计量单位定义为加权资源消耗，向上取整：

$$u(c) = \left\lceil w_g \cdot g(c) + w_s \cdot s(c) + w_x \cdot x(c) \right\rceil \quad (1)$$

g 为执行的逻辑门次数， s 为状态读写次数， x 为子电路或外部调用次数。权重由电路作者在注册时声明，并包含在 metaHash 中，调用方开会话前即可校验。会话 σ 的累计单位与应付金额为：

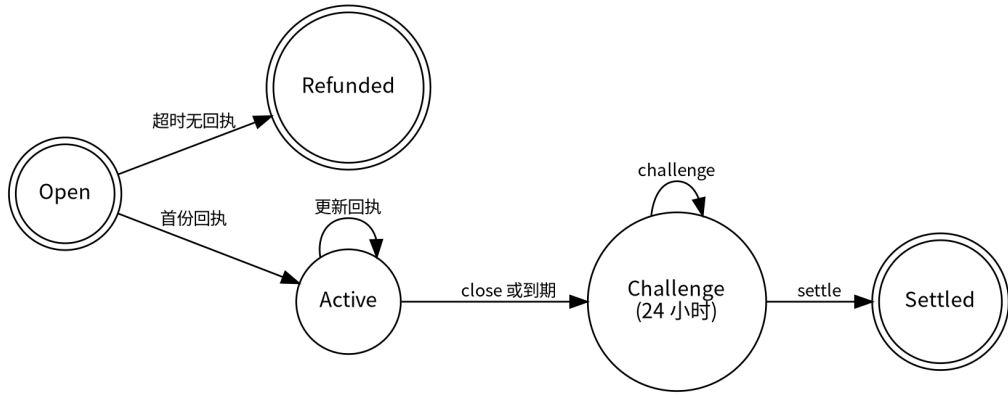
$$U_\sigma = \sum_{i=1}^n u(c_i), \quad G_\sigma = \min(U_\sigma \cdot p_\sigma, D_\sigma) \quad (2)$$

p_σ 为开会话时快照的单价， D_σ 为押金，调用方退款为 $D_\sigma - G_\sigma$ 。回执 r_k 有效当且仅当：

$$\text{nonce}_k > \text{nonce}_{last} \wedge \text{units}_k \geq \text{units}_{last} \wedge \text{now} \leq \text{deadline}_k \wedge \text{Sig}_{caller} \wedge \text{Sig}_{meter} \quad (3)$$

合约只保留 nonce 最大的有效回执。子电路调用计入父会话的 $x(c)$ ，同时子电路独立开会话结算，避免嵌套记账在单笔交易中级联失败。

结算流程与状态机



会话状态机

- 1. **开会话**：调用方授权结算币并调用 `openSession`，合约锁定押金并快照单价与分账参数。
- 2. **调用与联签**：网关每完成一批调用即生成新回执，调用方与计量方以 EIP-712 签名。
- 3. **提交**：任一方调用 `submitReceipt`，合约校验签名、nonce、deadline 与押金上限。
- 4. **挑战期**：会话关闭后进入 24 小时挑战期，持有更高 nonce 有效回执的任何人可调用 `challenge`。
- 5. **结算**：挑战期结束后任何人可调用 `settle`，合约按比例记账并退还剩余押金。
- 6. **提款**：收款方通过 `withdraw` 拉取余额；席位份额通过 `claimSeat` 凭 Merkle 证明领取。

收益分配

收款方	比例	依据
电路作者	60%	承担设计、流片与维护，是调用价值的主要来源
执行席位	25%	按实际工作量分配，激励席位持有者提供真实执行
计量方	10%	覆盖网关运行与联签成本，保证计量服务可持续
协议金库	5%	资助审计、挑战奖励与开发，比例保持较低以免挤占参与者

比例以基点表示（6000 / 2500 / 1000 / 500），合计 10000。整数除法的余数计入金库，保证不丢失、不超发：

$$a_j = \left\lfloor \frac{G_\sigma \cdot \text{bps}_j}{10000} \right\rfloor, \quad a_{\text{treasury}} = G_\sigma - \sum_{j \neq \text{treasury}} a_j \quad (4)$$

执行席位份额按回执 `seatsRoot` 中各席位的工作量 w_k 分配，未参与执行的席位不获得份额：

$$\text{seat}_k = \left\lfloor \frac{0.25 \cdot G_\sigma \cdot w_k}{\sum w} \right\rfloor \quad (5)$$

项目（示意计算）	金额
会话结算额 G_o	100 USDT
电路作者	60 USDT
执行席位（合计）	25 USDT
计量方	10 USDT
协议金库	5 USDT

合约接口

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.24;

/// TCS-1 电路注册表
interface ITCS1Registry {
    struct Circuit {
        address owner;           // 电路作者
        address processor;       // TapeOut Processor（晶体管合约）
        uint256 seatId;          // 席位类型：0 = NAND, 1 = LATCH
        uint256 unitPrice;       // 每计量单位价格（结算币最小单位）
        address payToken;        // 结算币（BEM 或 USDT）
        bytes32 metaHash;        // 电路元数据与计量权重的哈希
        uint32 version;
        bool active;
    }

    event CircuitRegistered(bytes32 indexed circuitId, address indexed owner,
        address processor, uint256 unitPrice, address payToken);
    event CircuitUpdated(bytes32 indexed circuitId, uint32 version,
        uint256 unitPrice, bytes32 metaHash);
    event CircuitActiveSet(bytes32 indexed circuitId, bool active);

    error NotCircuitOwner();
    error CircuitInactive();
    error UnsupportedToken(address token);

    function registerCircuit(address processor, uint256 seatId, uint256 unitPrice,
        address payToken, bytes32 metaHash) external returns (bytes32 circuitId);
    function updateCircuit(bytes32 circuitId, uint256 unitPrice, bytes32 metaHash)
        external; // 仅对新会话生效
    function setActive(bytes32 circuitId, bool active) external;
    function circuit(bytes32 circuitId) external view returns (Circuit memory);
}

/// TCS-1 会话结算
interface ITCS1Settlement {
```

```

enum State { None, Open, Active, Challenge, Settled, Refunded }

struct Receipt {          // EIP-712 计量回执
    bytes32 sessionId;
    bytes32 circuitId;
    address caller;
    uint64 units;          // 会话累计计量单位, 单调不减
    uint64 nonce;          // 严格递增
    uint64 deadline;
    bytes32 seatsRoot;     // 实际执行席位及其工作量的 Merkle 根
}

event SessionOpened(bytes32 indexed sessionId, bytes32 indexed circuitId,
    address indexed caller, uint256 deposit, uint256 unitPrice);
event ReceiptSubmitted(bytes32 indexed sessionId, uint64 units, uint64 nonce);
event SessionClosed(bytes32 indexed sessionId, uint64 challengeEnds);
event ReceiptChallenged(bytes32 indexed sessionId, address indexed challenger,
    uint64 nonce);
event SessionSettled(bytes32 indexed sessionId, uint256 gross, uint256 refund);
event Credited(bytes32 indexed sessionId, address indexed to, uint8 indexed role,
    uint256 amount);
event SeatClaimed(bytes32 indexed sessionId, address indexed seatHolder,
    uint256 amount);

error InvalidSignature();
error StaleNonce(uint64 got, uint64 last);
error UnitsDecreased();
error Expired();
error ExceedsDeposit(uint256 cost, uint256 deposit);
error WrongState(State state);
error ChallengeWindowOpen();
error InvalidProof();

function openSession(bytes32 circuitId, uint256 deposit, uint64 ttl)
    external returns (bytes32 sessionId);
function submitReceipt(Receipt calldata r, bytes calldata callerSig,
    bytes calldata meterSig) external;
function close(bytes32 sessionId) external;
function challenge(Receipt calldata newer, bytes calldata callerSig,
    bytes calldata meterSig) external;
function settle(bytes32 sessionId) external;          // 无需许可
function refundExpired(bytes32 sessionId) external;   // 超时且无回执
function claimSeat(bytes32 sessionId, address holder, uint256 work,
    bytes32[] calldata proof) external;
function withdraw(address token) external;            // pull 模式
}

/// 席位发行 (可选模块)
interface ITCS1SeatDistributor {
    event SeatsIssued(address indexed to, uint256 seatId, uint256 amount, uint8 phase);
    error NotEligible();
    error CapExceeded();
    error PhaseClosed();
    function issueAllowlist(uint256 seatId, uint256 amount, bytes32[] calldata proof)

```

```

    external;
    function issuePublic(uint256 seatId, uint256 amount) external;
}

```

安全模型

威胁	缓解措施
回执重放	EIP-712 域包含 chainId、合约地址与 sessionId；nonce 严格递增；回执带 deadline
计量方虚报	回执需调用方联签；支出以押金为上限；挑战期内可被更新回执覆盖
调用方拒签	网关按批次预签，未签批次不继续服务，单批为最大敞口
女巫与刷量	调用费用由调用方实付；席位份额仅按实际工作量分配
阻塞攻击	pull 模式提款；settle 与 refundExpired 无需许可
双方合谋	影响范围限于合谋者自身押金；电路作者可停用电路或更换计量方
结算币风险	仅支持白名单代币，使用 SafeERC20，不支持转账收税代币
合约升级	TCS-1 v1.0 合约不可升级；参数变更通过 48 小时时间锁

经济模型

- 电路单价由作者自定，由市场竞争决定，协议不设指导价。
- 会话越长，开关会话的固定 gas 成本摊薄越充分。
- 计量份额保证网关运营方有持续的经济动机。
- 金库支出用途公开，并在链上可查。
- CircuitPay 不发行协议代币，结算使用现有代币（BEM、USDT）。

与 TapeOut 的集成

CircuitPay 的席位基于 TapeOut 处理器发行。下列参数为 TapeOut 工厂合约的公开行为。

项目	说明
工厂合约	0x68224F668083c29e9800Be2a646d42d18cedF7e2
创建 Processor	createCPU(name, symbol, story, transistorSupply, mintPrice)，部署费 0.001 BNB
铸造	mint(id, amount)，费用为 amount × mintPrice 加每次调用 0.0001 BNB 协议费
铸造收入	记入 Processor 创建者，通过 withdraw() 提取
席位类型	id 0 = NAND，id 1 = LATCH

CircuitPay 通过 ITCS1SeatDistributor 发行席位，分为优先名单阶段和公开阶段。优先名单以 Merkle 根存储于链上，每个地址设有发行上限，支持以 BEM 或 USDT 支付。

风险披露

- 底层协议可升级：TapeOut 工厂合约 of UUPS 代理，由 3/5 多签管理，并可通过 beacon 升级其创建的晶体管合约。底层合约行为的变化可能影响席位所有权的读取，CircuitPay 会持续监控并在必要时暂停新电路注册。
- 智能合约风险：合约可能存在未发现的漏洞。
- 计量信任：计量在链下完成，安全性依赖双方签名与挑战机制。
- 结算币风险：结算代币的价格与可用性不受 CircuitPay 控制。
- 在自定义 Processor 上流片的电路不参与 TapeOut 官方 PoD 挖矿。
- 监管风险：各司法辖区对链上应用的监管可能变化。

路线图

1. 阶段一：测试网：部署 Registry、Settlement 与 SeatDistributor，发布参考网关。
2. 阶段二：主网：独立安全审计完成后上线 BSC 主网，开启席位发行。
3. 阶段三：开放计量：支持多个计量方，上线挑战奖励。
4. 阶段四：电路组合：支持子电路调用的批量结算。

术语表

术语	含义
CircuitPay	构建在 TapeOut 之上的电路调用结算层
TCS-1	CircuitPay 定义的电路调用结算标准
Processor	TapeOut 工厂创建的处理器合约
晶体管 / 席位	Processor 铸出的代币，在 CircuitPay 中代表一个 agent 席位
电路 / Swarm	由席位组成、协作执行任务的 agent 集群
流片	部署一个项目或电路
会话	一笔押金对应的一组调用及其结算
计量回执	调用方与计量方共同签名的累计计量凭证
挑战期	结算前允许提交更新回执的时间窗口

免责声明

本白皮书仅用于说明 CircuitPay 与 TCS-1 的技术设计，不构成投资建议、要约或任何形式的收益承诺。文中参数可能随协议演进而调整，以链上部署的合约为准。示意计算仅用于说明公式，不代表实际收入。参与者应自行评估风险并遵守所在地法律法规。