ZEC奖励计算智能合约设计文档

一、业务描述

zec奖励计算合约负责定时计算本周期的发行总量，从而计算出本周期的 edge奖励、archive 奖励、validator 奖励、auditor奖励、投票奖励、治理奖励。

最后根据节点承担的角色将上述类型的奖励（除治理奖励外）分配给节点，节点再根据设置的分红比例将节点奖励分配给投票者，zec 没有每个投票者的投票信息，

因此，zec 只计算分配到 table 合约的总的投票分红，再由table 奖励领取合约计算每个投票者应得分红。

二、详细设计

代码文件：

src/xtopcom/xvm/xsystem_contracts/xreward/xzec_reward_contract.h

src/xtopcom/xvm/xsystem_contracts/src/xzec_reward_contract.cpp

类名：

xzec_reward_contract

属性：

XPORPERTY_CONTRACT_TASK_KEY：保存任务列表

XPORPERTY_CONTRACT_TIME_KEY：保存上次触发定时处理函数的时钟高度

XPROPERTY_CONTRACT_ACCUMULATED_ISSUANCE：保存总发行量

XPROPERTY_CONTRACT_ACCUMULATED_ISSUANCE_YEARLY：保存按年统计的总发行量

相关数据结构

- xaccumulated_reward_record

保存top币累计发行量。

```
struct xaccumulated_reward_record final : public
xserializable_based_on<void> {
    common::xlogic_time_t last_issuance_time{0};
    uint64_t issued_until_last_year_end{0};
    uint64_t calculated_issuance{0};
private:
    /**
     * @brief           write to stream
     *
     * @param stream
     * @return int32_t
     */
    int32_t do_write(base::xstream_t & stream) const override;
    /**
     * @brief           read from stream
     *
     * @param stream
     * @return int32_t
     */
    int32_t do_read(base::xstream_t & stream) override;
};
```

字段	说明
last_issuance_time	上次发行时钟高度
issued_until_last_year_end	截止到去年年底时间发行总量，基本无用
calculated_issuance	调整后截止到去年年底时间发行总量，涉及到跨年发行时，需要补全到去年年底的发行量

- xreward_dispatch_task

保存分发任务，工3类任务，发行top币任务、下发节点自留奖励任务、下发节点分红任务

```
struct xreward_dispatch_task final : public xserializable_based_on<void> {
    uint64_t onchain_timer_round;
    std::string contract;
    std::string action;
    std::string params;


private:
    /**
     * @brief write to stream
     *
     * @param stream
     * @return int32_t
     */
    int32_t do_write(base::xstream_t & stream) const override {
        const int32_t begin_pos = stream.size();
        stream << onchain_timer_round;
        stream << contract;
        stream << action;
        stream << params;
        const int32_t end_pos = stream.size();
        return (end_pos - begin_pos);
    }

    /**
     * @brief read from stream
     *
     * @param stream
     * @return int32_t
     */
    int32_t do_read(base::xstream_t & stream) override {
        const int32_t begin_pos = stream.size();
        stream >> onchain_timer_round;
        stream >> contract;
        stream >> action;
        stream >> params;
        const int32_t end_pos = stream.size();
        return (begin_pos - end_pos);
    }
};
```

字段	说明
onchain_timer_round	本轮发行时钟高度
contract	合约账户
action	任务类型
params	任务参数

流程说明

1、计算奖励

a、计算本周期发行总量，本期发行总量为本年度预留奖励*本周期时钟数/年度总时钟数，

如果截止去年底已发行量小于总预留增发总额，则本年度预留奖励为剩余预留发行总额（总预留增发总额减截止去年底已发行量）乘以年度发行比率，

如果截止去年底已发行量大于等于总预留增发总额，则本年度预留奖励为系统定义的年度最小发行量，

如果上次发行时间在上年，则本次发行跨年，需要分两阶段计算，第一阶段计算上次发行时间至上年底发行量，第二阶段计算上年度至现在发行量，计算方法按上述算法。

b、计算出本期发行总量后，按预定比例分别计算出本周期的 edge奖励、archive 奖励、validator 奖励、auditor奖励、投票奖励、治理奖励

c、统计出 edge 节点数、archive节点数、auditor 节点数

d、如果节点是edge节点，节点edge奖励=edge奖励/edge 节点数；

如果节点是archive节点，节点archive奖励=archive 奖励/archive节点数；

如果节点是validator节点，节点validator奖励=∑validator 奖励/validator group数 * 节点的validator 工作量/所在group总工作量， 同一个节点有可能在多个 validator group 承担 validator 角色，

如果节点是创世节点，则这部分奖励划归公共账户，如果不是，则归节点自身；

如果节点是auditor节点，节点auditor奖励=∑validator 奖励/auditor group数 * 节点的auditor 工作量/所在group总工作量，同一个节点有可能在多个 auditor group 承担 auditor 角色，

如果节点是创世节点，则这部分奖励划归公共账户，如果不是，则归节点自身；

如果节点是auditor节点，节点投票奖励=投票奖励*节点所得票数/总票数，如果节点是创世节点，则这部分奖励划归公共账户，如果不是，则归节点自身，

根据设置的分红比例计算自留奖励，其余奖励分配给各table 合约，按照各table合约投本节点的票数占比分配，这里只分配到table合约，

因为zec 上没有投票者的详细投票数据，只有汇总数据；

e、如果某个validator group 没有工作量或者总工作量小于系统定义的阈值，则group 应得的 validator 奖励称为零工作量奖励，划归公共账户，

如果某个auditor group 没有工作量或者总工作量小于系统定义的阈值，则group 应得的 auditor 奖励称为零工作量奖励，划归公共账户。

2、生成分发奖励任务

分发奖励的目的是把节点自留奖励及节点分给投票者的分红下发到table奖励领取合约，为了控制每次共识交易数量，分发奖励生成3类任务，后台定时触发检查任务列表，每次取20个任务执行。3类任务分别为：向table合约发行top币任务、

向table 合约下发节点自留奖励任务、向table合约下发节点分红任务。

向table合约发行top币任务：第1步生成了本期发行币总量，需要将币发行到table 奖励领取合约，以便节点及投票者领取相应奖励和分红

向table 合约下发节点自留奖励任务：需要将节点自留奖励明细下发到所属ttable奖励领取合约，以便节点领取奖励

向table合约下发节点分红任务：需要将节点投票分红明细下发到table奖励领取合约，以便table奖励领取合约计算本table内每个投票者分红明细

a、遍历节点自留奖励map，按节点账户所属table归类，同时汇总发行到table合约的币总量

b、遍历第1步生成的table 节点分红map, 汇总发行到table合约的币总量

c、遍历第a、b步生成的table合约的币总量map, 生成发行top币任务，每个table 一个任务

d、遍历按table 分类后的节点自留奖励map，生成下发节点自留奖励任务，每个table 一个任务

e、遍历按table 分类后的节点分红map, 生成下发节点分红任务，每个table 一个任务

3、执行任务

a、后台定时检查任务列表，每次取20个任务执行