PDX Utopia

质押加入共识委员会与退出共识委员会

版本 1.1.0

目 录

[1. 设计初衷 1](#_Toc49286514)

[2. 质押与退款介绍 1](#_Toc1577236703)

[2.1 质押 1](#_Toc79128753)

[2.2 退款 1](#_Toc624574178)

[3. 惩罚机制 1](#_Toc1944452387)

[4. 质押流程图 2](#_Toc5128263)

[5.交易实例 3](#_Toc291370361)

[5.1 质押示例 3](#_Toc798942167)

[5.2 退款示例 4](#_Toc1753239725)

## 

# **1. 设计初衷**

为了极大程度的保证共识委员会的成员的稳定性，防止有恶意节点大量且频繁的进出委员会对Utopia共识算法机制产生影响，并且给予参与到共识委员当中的成员产生最大的收益。对恶意退出委员会的节点作出一定的限制的惩罚。因此引入质押机制。

# **2. 质押与退款介绍**

## 2.1 质押

用户当启动一个节点链接Utopia主网的时候，会累计活跃度，当活跃度满足一定条件的时候，这个节点拥有参与到共识机制并且称为共识委员会成员之一的机会，这个时候用户可以通过购买PDX token的方式，将token转账给这个节点的挖矿地址，然后 通过发起质押交易的方式，向Utopia质押合约发起转账，即调用质押合约的方式，也可以通过已拥有足够PDX币的账户发起质押交易，赋予需要参与到共识机制的节点地址质押标识的方式。质押地址是一个固定的十六进制地址，合约名称为hypothecation，转换成地址信息以后为0x04D2FBC88555e90F457F814819224460519e73A1，当用户发起质押交易， Utopia在接收到质押交易以后，会根据发起质押合约交易的账户地址进行校验。

需要说明的是: 进入共识委员会的质押金额是与节点数量正相关的，规则为共识委员会成员每增加128个成员，质押金额翻倍的规则。那么最初的质押金额为:10万个PDX币。Utopia鼓励用户进行质押，质押金额越多额外的分红奖励也会越多。

当用户的质押金额满足条件以后，质押交易的合约执行成功，该用户拥有进入委员会的资格，如果质押的金额不够，质押交易的合约会执行失败，用户可以通过增加质押金的方式重新发起质押。

## 2.2 退款

当用户由于个人原因，想主动退出共识委员会，取回质押金额的时候,Utopia允许用户进行解除质押操作。解除质押操作为一步完成:第一步，发送退回质押金的交易，即调用退回质押金的合约名称为**hypothecation**，转换成地址信息为**0x04D2FBC88555e90F457F814819224460519e73A1**,Utopia在接收到解除质押交易以后，会根据该账户地址的质押记录进行退款。如果退款账户在发起退款的时候，已不在共识委员会，或者需要被惩罚(惩罚办法稍后介绍)，解除质押的合约会对该账户惩罚以后，将剩余的质押金退还

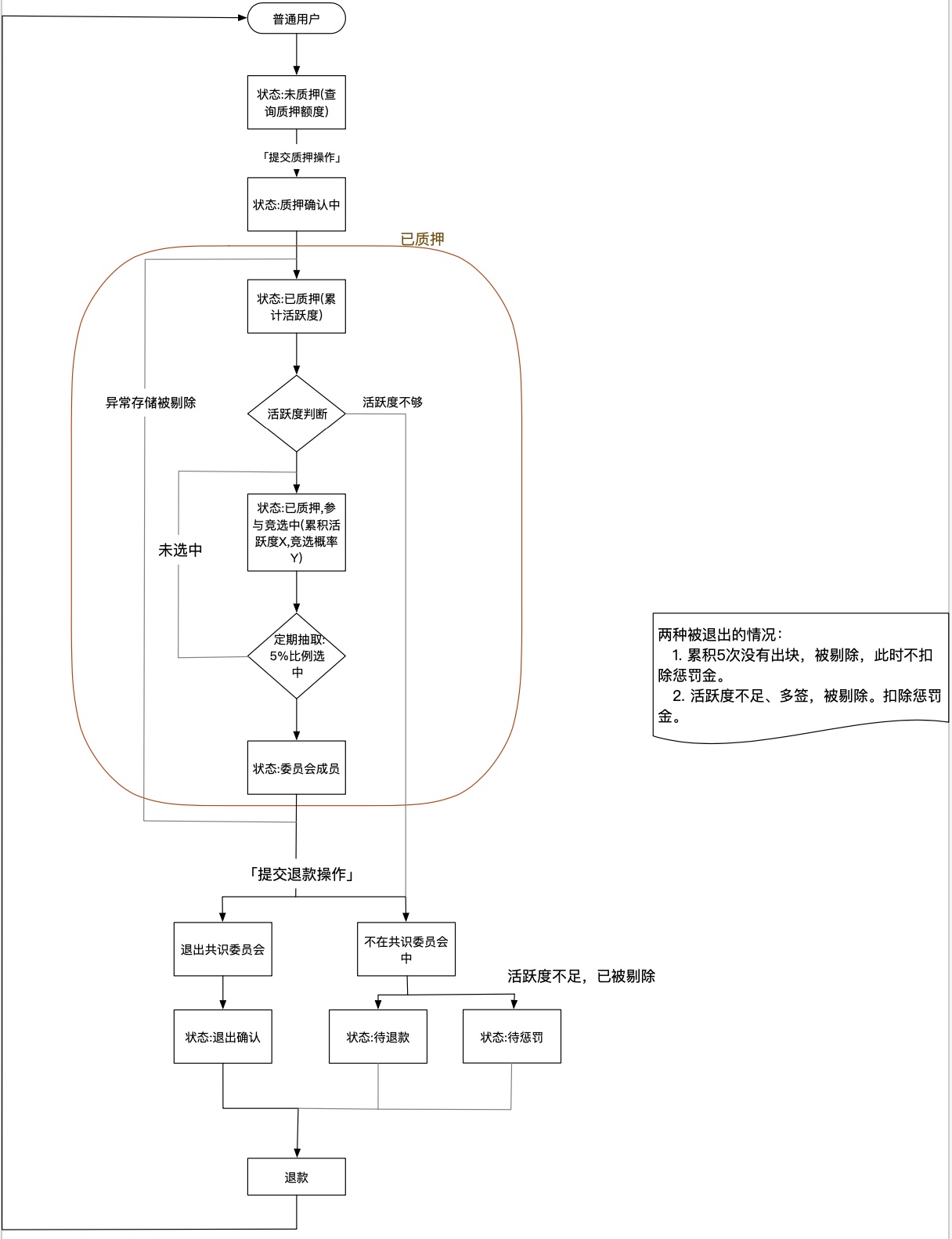
# **3. 惩罚机制**

为了让参与到共识机制的成员中少数恶意节点作恶有所顾忌，以及增加作恶成本。Utopia引入惩罚机制。

惩罚对象: 进行过质押交易的账户并且在连续几个区间内，没有发送assertion，而导致活跃度不足，被踢除出委员会的;多签被其他节点举报的;没有按时出块，或由于其他原因导致大部分节点没有收到其发送出的区块的。

惩罚措施: 扣除该账户初次质押金额的百分之十，作为惩罚。

# **4. 质押流程图**

****

# 5.交易实例

## 5.1 质押示例

|  |
| --- |
| import (  "context"  "fmt"  "math/big"  "pdx-chain/common"  "pdx-chain/core/types"  "pdx-chain/crypto"  "pdx-chain/p2p/discover"  "pdx-chain/rlp"  "pdx-chain/utopia/utils/client"  "testing"  )  func TestHypothecation\_Run(t \*testing.T) {  if client, err := client.Connect("http://127.0.0.1:8545"); err != nil {  fmt.Printf(err.Error())  return  } else {  if privKey, err := crypto.HexToECDSA("1263354aca47fe21d54e0fd4d125e879142a1367e6962bcdc8dd251830f4b092"); err != nil {  fmt.Printf(err.Error())  return  } else {  nodeID := discover.PubkeyID(&privKey.PublicKey)  from := crypto.PubkeyToAddress(privKey.PublicKey)  //质押  to := common.BytesToAddress(crypto.Keccak256([]byte("hypothecation"))[12:])  fmt.Printf("to:%s\n", to.String())  fmt.Println(from.String())  if nonce, err := client.EthClient.NonceAt(context.TODO(), from, nil); err != nil {  fmt.Printf("ddddd:%s", err.Error())  return  } else {  amount := big.NewInt(3000000000000000000)  gasLimit := uint64(4712388)  gasPrice := big.NewInt(240000000000) //todo 此处很重要，不可以太低，可能会报underprice错误，增大该值就没有问题了  type HypothecationAddress struct {  Address common.Address `json:"address"` //抵押地址  }  //addr := common.HexToAddress("0x41b84E81Ba73ab6910ac9b2038d6F18C15a05367")  recaption := &HypothecationAddress{  from,  }  payload, \_ := rlp.EncodeToBytes(recaption)  tx := types.NewTransaction(nonce, to, amount, gasLimit, gasPrice, payload)  //EIP155 signer  signer := types.NewEIP155Signer(big.NewInt(739)) signedTx, \_ := types.SignTx(tx, signer, privKey)  If txHash, err := client.SendRawTransaction(context.TODO(), signedTx); err != nil {  fmt.Println("yerror", err.Error())  } else {  fmt.Printf("Transaction hash: %s\n", txHash.String())  }  }  }  }  } |

## 5.2 退款示例

|  |
| --- |
| import (  "context"  "fmt"  "math/big"  "pdx-chain/common"  "pdx-chain/core/types"  "pdx-chain/crypto"  "pdx-chain/log"  "pdx-chain/p2p/discover"  "pdx-chain/rlp"  "pdx-chain/utopia/utils/client"  "testing"  )  func TestHypothecation\_Run2(t \*testing.T) {  if client, err := client.Connect("http://127.0.0.1:8545"); err != nil {  fmt.Printf(err.Error())  return  } else {  if privKey, err := crypto.HexToECDSA("1263354aca47fe21d54e0fd4d125e879142a1367e6962bcdc8dd251830f4b092"); err != nil {  fmt.Printf(err.Error())  return  } else {  nodeID := discover.PubkeyID(&privKey.PublicKey)  fmt.Printf("nodeID:%s \n", nodeID.String())  from := crypto.PubkeyToAddress(privKey.PublicKey)  fmt.Printf("from:%s\n", from.String())  //退钱  to := common.BytesToAddress(crypto.Keccak256([]byte("hypothecation"))[12:])  fmt.Printf("to:%s\n", to.String())  fmt.Println(from.String())  if nonce, err := client.EthClient.NonceAt(context.TODO(), from, nil); err != nil {  return  } else {  amount := big.NewInt(3000000000000000000)  gasLimit := uint64(4712388)  gasPrice := big.NewInt(240000000000) //todo 此处很重要，不可以太低，可能会报underprice错误，增大该值就没有问题了  ////退钱需要填充的  type RecaptionInfo struct {  HypothecationAddr common.Address `json:"hypothecation\_addr"` //质押的地址  RecaptionAddress common.Address `json:"recaption\_address"` //退款地址  }  /\*  对于退款地址可以通过以下方法设置  addr := common.HexToAddress("0x41b84e81ba73ab6910ac9b2038d6f18c15a05367"  \*/  recaption := &RecaptionInfo{  from,  from, //addr  }  payload, err := rlp.EncodeToBytes(recaption)  log.Info("payload", "payload", string(payload))  if err != nil {  fmt.Printf("marshal XChainTransferWithdraw err:%s", err)  return  }  tx := types.NewTransaction(nonce, to, amount, gasLimit, gasPrice, payload)  //EIP155 signer  signer := types.NewEIP155Signer(big.NewInt(773))  signedTx, \_ := types.SignTx(tx, signer, privKey)  if txHash, err := client.SendRawTransaction(context.TODO(), signedTx); err != nil {  fmt.Println("yerror", err.Error())  } else {  fmt.Printf("Transaction hash: %s\n", txHash.String())  }  }  }  }  } |