PAXG节点质押系统/流动性/dapp智能合约/详情

产品名称	PAXG节点质押系统/流动性/dapp智能合约/详情
公司名称	广州杰肯狸网络科技有限公司
价格	.00/件
规格参数	
公司地址	广州天河区中山大道
联系电话	18125913365 19927739756

产品详情

通俗地说，可以把***比作一种“ 账本 ”。传统账本由一方“ 集中记账 ”，这种新式“ 账本 ”【系统-176搭建-0206可电可微-5616】则可以在互联网上由多方参与、共享，各参与方都可以“ 记账 ”并备份，而每个备份就是一个“ 区块 ”。每个“ 区块 ”与下一个“ 区块 ”按时间顺序线性相连，其结构特征使记录无法被篡改和伪造。

（Blockchain）是由节点参与的分布式数据库系统， 也可以将其理解为账簿系统(ledger)，其实就是用来记账，用来记录每一笔交易的，且能保证每一笔交易记录公开透明，不可篡改伪造。由于是去中心化的网络，在上的每一笔交易都需要“ 矿工 ”去“ 挖矿 ”来记下这笔帐，存储到链上去。

以太坊采用了Solidity作为智能合约语言，176开阀0206模式5616,Solidity是一门为实现智能合约而创建的gao.级编程语言，能在允许以太坊程序的节点上运行。该语言吸收了C++、JavaScript的一些特性，例如它是静态类型语言，支持继承库等。

//fetches and sorts the reserves for a pair

function getReserves(address factory,address tokenA,address tokenB)internal view returns(uint reserveA,uint reserveB){

(address token0,)=sortTokens(tokenA,tokenB);

```
(uint reserve0,uint reserve1,)=IUniswapV2Pair(pairFor(factory,tokenA,tokenB)).getReserves();

(reserveA,reserveB)=tokenA==token0?(reserve0,reserve1):(reserve1,reserve0);

}

//given some amount of an asset and pair reserves,returns an equivalent amount of the other asset

function quote(uint amountA,uint reserveA,uint reserveB)internal pure returns(uint amountB){

require(amountA>0,'UniswapV2Library:INSUFFICIENT_AMOUNT');

require(reserveA>0&&reserveB>0,'UniswapV2Library:INSUFFICIENT_LIQUIDITY');

amountB=amountA.mul(reserveB)/reserveA;

}

//given an input amount of an asset and pair reserves,returns the maximum output amount of the other asset

function getAmountOut(uint amountIn,uint reserveIn,uint reserveOut)internal pure returns(uint amountOut){

require(amountIn>0,'UniswapV2Library:INSUFFICIENT_INPUT_AMOUNT');

require(reserveIn>0&&reserveOut>0,'UniswapV2Library:INSUFFICIENT_LIQUIDITY');
```

```
uint amountInWithFee=amountIn.mul(997);

uint numerator=amountInWithFee.mul(reserveOut);

uint denominator=reserveIn.mul(1000).add(amountInWithFee);

amountOut=numerator/denominator;

}

//given an output amount of an asset and pair reserves,returns a required input amount of the other asset

function getAmountIn(uint amountOut,uint reserveIn,uint reserveOut)internal pure returns(uint amountIn){

require(amountOut>0,'UniswapV2Library:INSUFFICIENT_OUTPUT_AMOUNT');


uint numerator=reserveIn.mul(amountOut).mul(1000);

uint denominator=reserveOut.sub(amountOut).mul(997);

amountIn=(numerator/denominator).add(1);

}
```

```
//performs chained getAmountOut calculations on any number of pairs

function getAmountsOut(address factory,uint amountIn,address[]memory path)internal view returns(uint[]memory amounts){

require(path.length>=2,'UniswapV2Library:INVALID_PATH');

amounts=new uint[](path.length);

amounts[0]=amountIn;

for(uint i;i<path.length-1;i++){

(uint reserveIn,uint reserveOut)=getReserves(factory,path<i>,path[i+1]);

amounts[i+1]=getAmountOut(amounts<i>,reserveIn,reserveOut);

}

}

//performs chained getAmountIn calculations on any number of pairs

function getAmountsIn(address factory,uint amountOut,address[]memory path)internal view returns(uint[]memory amounts){
```

```
amounts=new uint[](path.length);

amounts[amounts.length-1]=amountOut;

for(uint i=path.length-1;i>0;i--){

(uint reserveIn,uint reserveOut)=getReserves(factory,path[i-1],path<i>);

amounts[i-1]=getAmountIn(amounts<i>,reserveIn,reserveOut);

}

}
```