拉撒路 API

V0.1

目录

1 历史

本节记录本文档的修改历史

版本	日期	作者	内容
v0.1	2020年4月23日	时空大神	初版本

2 JSON RPC

JSON RPC 的详细描述

方法	用途	内容
getBalance	获取余额	`{{"jsonrpc":"2.0","id":1,"method":"getBalance","params":["{}"]}}`
getTransactionCount	获取交易数量	{{"jsonrpc":"2.0","id":1,"method":"getTransactionCount"}}
getAccountInfo	获取账户信息	{{"jsonrpc":"2.0","id":1,"method":"getAccountInfo","params":["{}"]}}
getSignatureStatus	获取交易状态	{{"jsonrpc":"2.0","id":1,"method":"getSignatureStatus","params":["{}"]}}

3 API 接口

本节详细说明已有的相关接口

方法	用途	参数	内容

`request_airdrop`	空投	`drone_addr: &SocketAddr`
		`id: &Pubkey`
		`tokens: u64`	

atomic_new	创建转账交易	from_keypair: &Keypair, to: Pubkey, tokens: i64, last_id: Hash	
send_with_sig	发送交易	tx: &Transaction	

request_airdrop

```
request_airdrop(
    drone_addr: &SocketAddr,
    id: &Pubkey,
    tokens: u64
);
```

atomic_new

```
atomic_new (
  from_keypair: &Keypair,
  to: Pubkey,
  tokens: i64,
  last_id: Hash
)
```

send_with_sig

```
send_with_sig (
      tx: &Transaction
```

			);
tx_cntr	获取交易数量		tx_cntr（）

transaction_count	计算 TPS	tx_count
		duration.subsec_nanos()
		duration.as_secs()
			let ns = duration.as_secs() * 1_000_000_000 + u64::from(duration.subsec_nanos());
			let tx_count = client.transaction_count();
			let tps = (tx_count * 1_000_000_000) as f64 / ns as f64;

4 接口引用

本节详细说明各类接口的使用方法

编号	用途	方法	返回	参数	文档引用
1	创建一个交易	tbkchain-baseline:: AtomicTx:: atomic_move (from_keypair: &Keypair, to: Pubkey, tokens: i64, last_id: Hash, fee: i64);	交易签名；	from_keypair id: &Pubkey tokens: u64	N/A
2	对交易签名进行验证	tbkchain-baseline::sigverify:: tbk25519_verify (batches: &[SharedPackets])；	Boolean	(batches: &[SharedPackets]	N/A

3	账户余额	tbkchain-baseline::sigverify::balance_of(pubkey: &Pubkey)；	账户余额	pubkey: &Pubkey	N/A
4	合约验证	test_transfer_on_date(from_pubkey, to_pubkey, dt, tokens); tbkchain-baseline::BudgetTransaction ::budget_new_on_date (from_keypair: &Keypair, to: Pubkey, contract: Pubkey, dt: DateTime<Utc>, dt_pubkey: Pubkey, cancelable: Option<Pubkey>, tokens: i64, last_id: Hash);	合约状态，账户余额	from_pubkey, to_pubkey, dt, tokens from_keypair: &Keypair, to: Pubkey, contract: Pubkey, dt: DateTime<Utc>, dt_pubkey: Pubkey, cancelable: Option<Pubkey>, tokens: i64, last_id: Hash	N/A
5	交易队列	register_entry_id(last_id: &Hash);	Err(LastIdNotFound)	last_id: &Hash	N/A
6	节点通信地址验证	is_valid_address(addr: &SocketAddr);	Boolean	addr: &SocketAddr	N/A
7	节点信息维护	verify_req_update_alive();	在线节点 ID 号		N/A
8	数据编码	serialize<T: ?Sized>(value: &T);；	字节流	value: &T	N/A
9	数据解码	serialize<T: ?Sized>(value: &T);	交易信息	(value: &T	N/A

10	广播交易	multicast (crdt: &Arc<RwLock<Crdt>>, leader_rotation_interval: u64, me: &NodeInfo, broadcast_table: &[NodeInfo], window: &SharedWindow, s: &UdpSocket, transmit_index: &mut WindowIndex, received_index: u64);	投票节点接收到的交易信息	crdt: &Arc<RwLock<Crdt>>, leader_rotation_interval: u64, me: &NodeInfo, broadcast_table: &[NodeInfo], window: &SharedWindow, s: &UdpSocket, transmit_index:&mut WindowIndex, received_index: u64	N/A
11	节点数据传输	acceptor (sock: Arc<UdpSocket>, exit: Arc<AtomicBool>, packet_sender: PacketSender, sender_tag: &'static str)；	接收方接收数据大小	sock: Arc<UdpSocket>, exit: Arc<AtomicBool>, packet_sender: PacketSender, sender_tag: &'static str	N/A
12	账本同步	book_keeping（）；	节点账本最新交易 ID 号；		N/A
13	投票验证	submit_nexus_vote (id: &Pubkey, keypair: &Keypair, bank: &Arc<Bank>, crdt: &Arc<RwLock<Crdt>>, vote_blob_sender: &BlobSender, last_vote: &mut u64, last_valid_validator_timestamp: &mut u64	Boolean	Id: 公钥地址； Keypair： 公私钥对； Bank： 交易数据； Crdt： 节点信息； last_vote： 上一次投票； last_valid_validator_timestamp：投票节点更新时间；	N/A

		);			
14	交易安全验证	verify(&self, start_hash: &Hash)；	Boolean	start_hash： 交易哈希；	N/A
15	交易信息同步	tbkchain-baseline:: doubling_phase:: DoublePhase；	最新账本状态		N/A
16	获取账户余额接口	balance_of(pubkey: Pubkey)；	账户余额	pubkey: 账户公钥；	N/A
17	获取交易状态接口	sign_rc(signature:Signature) (signature: Signature)；	交易状态；	tx_signature: 交易签名；	N/A