TRON Protobuf协议文档

TRON使用Google protobuf协议，协议内容涉及到账户，区块，传输多个层面。

- 账户有基本账户、资产发布账户和合约账户三种类型。一个账户包含，账户名称，账户类型，地址，余额，投票，其他资产6种属性。
- 更进一步的，基本账户可以申请成为验证节点，验证节点具有额外的属性，投票统计数目，公钥，URL，以及历史表现等参数。
- 3种Account类型：Normal，AssetIssue，Contract。

```
enum AccountType {
  Normal = 0;
  AssetIssue = 1;
  Contract = 2;
}
```

- 一个Account包含6种参数：
account_name：该账户的名称——比如：*“BillsAccount”*。
type:该账户的类型——比如： 0代表的账户类型是Normal。
balance:该账户的TRX余额——比如：4213312。
votes:账户所得投票数——比如：*{(“0x1b7w…9xj3”,323),(“0x8djq…j12m”,88),…,(“0x82nd…mx6i”,10001)}* 。
asset：除TRX以外账户上的其他资产——比如：*{<”WishToken”,66666>,<”Dogie”,233>}* 。

```
// Account
message Account {
  message Vote {
    bytes vote_address = 1;
    int64 vote_count = 2;
  }
  bytes accout_name = 1;
  AccountType type = 2;
  bytes address = 3;
  int64 balance = 4;
  repeated Vote votes = 5;
  map<string, int64> asset = 6;
}
```

一个Witness包含8种参数：
address：验证节点的地址——比如：*“0xu82h…7237”*。
voteCount：验证节点所得投票数——比如：*234234* 。
pubKey：验证节点的公钥——比如: *“0xu82h…7237”*。
url：验证节点的url链接——比如：*“https://www.noonetrust.cn”*。
totalProduce：验证节点产生的区块数——比如：*2434*。
totalMissed：验证节点丢失的区块数——比如：*7*。
latestBlockNum：最新的区块高度——比如：*4522* 。

```
// Witness
message Witness {
  bytes address = 1;
  int64 voteCount = 2;
  bytes pubKey = 3;
  string url = 4;
  int64 totalProduced = 5;
  int64 totalMissed = 6;
  int64 latestBlockNum = 7;
}
```

一个区块由区块头和多笔交易构成。区块头包含时间戳，交易字典树的根，父哈希，签名等区块基本信息。

一个block包含transactions和block_header。
transactions：区块里的交易信息。
block_header：区块的组成部分之一。

```
// block
message Block {
    repeated Transaction transactions = 1;
    BlockHeader block_header = 2;
```

BlockHeader 包括raw_data和witness_signature。
raw_data：raw信息。
witness_signature：区块头到验证节点的签名。

message raw包含6种参数：
timestamp：该消息体的时间戳——比如：*14356325* 。
txTrieRoot：Merkle Tree的根——比如：*“7dacsa…3ed”*。
parentHash：上一个区块的哈希值——比如 ：*“7dacsa…3ed”*。
number：区块高度——比如 ：*13534657* 。
witness_id：验证节点的id——比如：*“0xu82h…7237”*。
witness_address：验证节点的地址——比如：*“0xu82h…7237”*。

```
message BlockHeader {
message raw {
    int64 timestamp = 1;
    bytes txTrieRoot = 2;
    bytes parentHash = 3;
    //bytes nonce = 5;
    //bytes difficulty = 6;
    uint64 number = 7;
    uint64 witness_id = 8;
    bytes witness_address = 9;
    }
    raw raw_data = 1;
    bytes witness_signature = 2;
   }
```

- 交易合约有多种类型，包括账户创建合约、转账合约、转账断言合约、资产投票合约、见证节点投票合约、见证节点创建合约、资产发布合约、部署合约8种类型。

AccountCreatContract包含3种参数：
type：账户类型——比如：0代表的账户类型是*transfer contract*。
account_name: 账户名称——比如: *“Billsaccount”*。
owner_address:合约持有人地址——比如：*“0xu82h…7237”*。

```
message AccountCreateContract {
    AccountType type = 1;
    bytes account_name = 2;
    bytes owner_address = 3;
}
```

TransferContract包含3种参数：
amount：TRX数量——比如：*12534* 。
to_address： 接收方地址——比如：*“0xu82h…7237”*。
owner_address：合约持有人地址——比如：*“0xu82h…7237”*。

```
message TransferContract {
    bytes owner_address = 1;
    bytes to_address = 2;
    int64 amount = 3;
}
```

TransferAssetContract包含4种参数：
asset_name：资产名称——比如：*“Billsaccount”*。
to_address：接收方地址——比如：*“0xu82h…7237”* 。
owner_address：合约持有人地址——比如：*“0xu82h…7237”*。
amount：目标资产数量——比如：*12353* 。

```
message TransferAssetContract {
    bytes asset_name = 1;
    bytes owner_address = 2;
    bytes to_address = 3;
    int64 amount = 4;
}
```

VoteAssetContract包含4种参数：
vote_address：投票人地址——比如：*“0xu82h…7237”* 。
support：投票赞成与否——比如：*true*。
owner_address：合约持有人地址——比如：*“0xu82h…7237”* 。
count：投票数目——比如：*2324234* 。

```
message VoteAssetContract {
    bytes owner_address = 1;
    repeated bytes vote_address = 2;
    bool support = 3;  int32 count = 5;
}
```

VoteWitnessContract包含4种参数：
vote_address：投票人地址——比如：*“0xu82h…7237”* 。
support：投票赞成与否——比如：*true* 。
owner_address：合约持有人地址——比如：*“0xu82h…7237”*。
count：投票数目——比如：*32632* 。

```
message VoteWitnessContract {
    bytes owner_address = 1;
    repeated bytes vote_address = 2;
    bool support = 3;
    int32 count = 5;
}
```

WitnessCreateContract包含3种参数：
private_key：合约的私钥——比如：*“0xu82h…7237”* 。
owner_address：合约持有人地址——比如：*“0xu82h…7237”*。
url：合约的url链接——比如：*“https://www.noonetrust.cn”*。

```
message WitnessCreateContract {
    bytes owner_address = 1;
    bytes private_key = 2;
    bytes url = 12;
}
```

AssetIssueContract包含11种参数：
name：合约名称——比如：*“billscontract”*。
total_supply：合约的赞成总票数——比如：*100000000* 。
owner_address：合约持有人地址——比如：*“0xu82h…7237”*。
trx_num：对应TRX数量——比如：*232241*。
num： 对应的自定义资产数目
start_time：开始时间——比如：*20170312*。
end_time：结束时间——比如：*20170512* 。
decav_ratio：衰减速率
vote_score：合约的评分——比如：*12343* 。
description：合约的描述——比如: *“trondada”* 。
url：合约的url地址链接——比如：*“https://www.noonetrust.cn”*。

```
message AssetIssueContract {
    bytes owner_address = 1;
    bytes name = 2;  int64 total_supply = 4;
    int32 trx_num = 6;  int32 num = 8;
    int64 start_time = 9;
    int64 end_time = 10;
    int32 decay_ratio = 15;
    int32 vote_score = 16;
    bytes description = 20;
    bytes url = 21;
}
```

DeployContract包含2种参数：
script：脚本。
owner_address：合约持有人地址——比如：*“0xu82h…7237”*。

```
message DeployContract {
    bytes owner_address = 1;
    bytes script = 2;
}
```

每一个交易还包含多个输入与多个输出，以及其他一些相关属性。其中交易内的输入，交易本身，区块头均需签名。

message Transaction包括raw_data和signature。
raw_data: 消息体raw。
signature: 所有输入节点的签名。

raw_data包含7种参数：
type：消息体raw的交易类型。
vin： 输入值
vout： 输出值
expiration：过期时间——比如：20170312。
data： 数据
contract： 该交易内的合约
script： 脚本

message Contract包含type和parameter。
type：合约的类型
parameter：任意参数

有八种账户类型合约：AccountCreateContract，TransferContract，TransferAssetContract，VoteAsset-Contract，VoteWitnessContract，WitnessCreateContract，AssetIssueContract 和DeployContract。

TransactionType包括UtxoType和ContractType。

```
message Transaction {
    enum TranscationType {
        UtxoType = 0;
        ContractType = 1;
}
message Contract {
    enum ContractType {
        AccountCreateContract = 0;
        TransferContract = 1;
        TransferAssetContract = 2;
        VoteAssetContract = 3;
        VoteWitnessContract = 4;
        WitnessCreateContract = 5;
        AssetIssueContract = 6;
        DeployContract = 7;
 }
 ContractType type = 1;
 google.protobuf.Any parameter = 2;
 }
 message raw {
     TranscationType type = 2;
     repeated TXInput vin = 5;
     repeated TXOutput vout = 7;
     int64 expiration = 8;
     bytes data = 10;
     repeated Contract contract = 11;
     bytes scripts = 16;
 }
 raw raw_data = 1;
repeated bytes signature = 5;
}
```

message TXOutputs由outputs构成。
outputs: 元素为TXOutput的数组。

```
message TXOutputs {
 repeated TXOutput outputs = 1;
}
```

message TXOutput包括value和pubKeyHash。
value：输出值。
pubKeyhash：公钥的哈希。

```
message TXOutput {
    int64 value = 1;
    bytes pubKeyHash = 2;
}
```

message TXIutput包括raw_data和signature。
raw_data：消息体raw。
signature：TXInput的签名。

message raw包含txID，vout和 pubKey。
txID：交易ID。
Vout：上一个输出的值。
pubkey:公钥。

```
message TXInput {
    message raw {
        bytes txID = 1;
        int64 vout = 2;
        bytes pubKey = 3;
}
raw raw_data = 1;
bytes signature = 4;
}
```

- 传输涉及的协议Inventory主要用于传输中告知接收方传输数据的清单。

Inventory包括type和ids。
type：清单类型——比如：0代表TRX。
ids：清单中的物品ID。

InventoryType包含TRX和 BLOCK。
TRX：交易。
BLOCK：区块。

```
// Inventory
message Inventory {
enum InventoryType {
 TRX = 0;    BLOCK = 1;
 }
 InventoryType type = 1;
repeated bytes ids = 2;
}
```

message Items包含4种参数：
type：物品类型——比如：1 代表 TRX.
blocks：物品中区块。
blockheaders：区块头。
transactions：交易。

Items有四种类型，分别是 ERR， TRX，BLOCK 和BLOCKHEADER。
ERR：错误。
TRX：交易。
BLOCK：区块。
BLOCKHEADER：区块头。

```
message Items {
enum ItemType {
 ERR = 0;
 TRX = 1;
 BLOCK = 2;
 BLOCKHEADER = 3;
 }
 ItemType type = 1;
repeated Block blocks = 2;
repeated BlockHeader block_headers = 3;
repeated Transaction transactions = 4;
}
```

Inventory包含type和items。
type：物品种类。
items：物品清单。

```
message InventoryItems {
  int32 type = 1;
 repeated bytes items = 2;
}
```

钱包服务RPC

Wallet钱包服务包含多个RPC。
Getbalance：获取Account的余额。
CreatTransaction：通过TransferContract创建交易。
BroadcastTransaction：广播Transaction。
CreateAccount：通过AccountCreateContract创建账户。
CreatAssetContract：通过AssetIssueContract发布一个资产。

```
service Wallet {
   rpc GetBalance (Account) returns (Account) {
   };
   rpc CreateTransaction (TransferContract) returns
(Transaction) {
   };
   rpc BroadcastTransaction (Transaction) returns (Return) {
   };
   rpc CreateAccount(AccountCreateContract) returns
(Transaction) {
   };
   rpc CreateAssetIssue(AssetIssueContract) returns
(Transaction) {
   };
};
```

消息体Return只含有一个参数:
result: 布尔表类型标志位。

```
message Return {
 bool result = 1;
}
```

详细的协议见附属文件。详细协议随着程序的迭代随时都可能发生变化，请以最新的版本为准。