**创建钱包地址**

MATRIX 主链钱包地址生成规则

1. 由secp256k1曲线生成私钥，是由随机的256bit组成
2. 采用椭圆曲线数字签名算法（ECDSA）将私钥映射成公钥
3. 公钥经过Keccak-256单向散列函数变成了256bit，然后取160bit作为Base58编码
4. 添加MAN.前缀，然后做CRC8 校验，生成一个结尾字符。

交易所需要创建一个在线钱包管理用户充值地址。钱包是用来存储账户（包含公钥和私钥）、合约地址等信息，是用户持有资产的最重要的凭证，一定要保管好keystore文件和钱包密码，不要丢失或泄露。

交易所不需要为每个地址创建一个钱包文件，使用一个冷钱包（离线钱包）是一种更安全的存储方式。

Matrix提供2种钱包地址创建方法。

1. 本地gman客户端创建。本地部署节点成功后(部署节点请参考：<https://github.com/MatrixAINetwork/MATRIX_docs/blob/master/%E4%B8%AD%E6%96%87%E6%96%87%E6%A1%A3/gman%E6%8A%B5%E6%8A%BC%E8%8A%82%E7%82%B9%E9%83%A8%E7%BD%B2%E6%89%8B%E5%86%8C%20(windows).pdf> 和 <https://github.com/MatrixAINetwork/MATRIX_docs/blob/master/%E4%B8%AD%E6%96%87%E6%96%87%E6%A1%A3/gman%E6%8A%B5%E6%8A%BC%E8%8A%82%E7%82%B9%E9%83%A8%E7%BD%B2%E6%89%8B%E5%86%8C(linux%26Mac).pdf> )，可以通过“personal.newAccount”api创建钱包地址，keystore文件存储在 <gman path>\chaindata\keystore路径下
2. 官方web钱包（wallet.matrix.io）或app钱包

**用户资产交易**

**查询**

交易所可以内部管理用户的余额，用户提现到个人钱包操作时，用户提现金额上链。

查询用户地址资产余额，交易所需要调用“man.getBalance” api获取用户地址余额

**充值**

关于用户充值，交易所需要了解以下内容：

* Matrix区块链只有一条主链，没有侧链，不会分叉。
* 所有记录在Matrix区块链中的交易都是不可篡改的，即一个确认就代表充值成功。
* Matrix钱包地址中不仅包含MAN币资产，还可以有许多种用户自己发行的全资产（如股权、Token等），交易所记录用户充值时需要判断充值资产的资产类型，以免把其它资产的充值当成MAN币，或把MAN币和Token的充值弄混。
* Matrix普通节点要保持在线才能同步区块，可以通过gman客户端的“man.blockNumber”api查看区块同步高度。
* 交易所内的用户之间转账不需要通过区块链，而可以直接修改数据库中的用户余额进行，只有充值提现才上链。

**充值记录**

交易所需要写代码监控每个区块的每个交易，在数据库中记录下所有充值提现交易。如果有充值交易就要修改数据库中的用户余额。

gman API 中的 man.getblock(Number) 方法提供了获取区块信息的功能，该方法中的Number为区块高度。

获取的区块信息中包含了交易详细信息，交易所需要记录下所有和自己相关的交易，作为用户充值提现的交易记录。如果发现在交易的输出中有属于交易所的地址，则要修改数据库中该充值地址对应的用户MAN币或Token余额。

也有交易所采用另一种方式：如果发现在交易的输出中有属于交易所的地址，先在数据库中记录下充值记录，待几个确认后再修改用户余额。如果不是为了与其它区块链操作方式统一，并不推荐这么做。

**交易示例**

Matrix提供js和go版本API库（AIMan），交易发送有以下几个步骤：

1. 用户keystore文件密码解锁获取私钥或直接输入私钥
2. 获取交易发送账户的序号nonce
3. 构建交易对象，注意区分普通交易和合约交易
4. 签名交易对象
5. 发送签名后的交易数据
6. 获取交易回执，确认交易上链状态并查询账户余额

**附录**

GO 语言api接口

|  |  |
| --- | --- |
| api名称 | BlockNumber |
| api描述 | 获取当前区块高度 |
| 输入参数 | 无 |
| 返回值 | err：错误信息  blocknumber：当前高度 |
| 示例 | blockNumber,err :=AIMan.GetBlockNumber() |

|  |  |
| --- | --- |
| api名称 | GetBlockByNumber |
| api描述 | 获取指定高度的区块信息 |
| 输入参数 | number \*big.Int, 要获取的指定区块的高度  transactionDetails bool 是否获取全部交易 |
| 返回值 | err：错误信息  block：指定高度的区块 |
| 示例 | block,err:=connection.Man.GetBlockByNumber(big.NewInt(211),false) |

|  |  |
| --- | --- |
| api名称 | GasPrice |
| api描述 | 获取当前GasPrice值 |
| 输入参数 | 无 |
| 返回值 | err：错误信息  blocknumber：当前高度 |
| 示例 | gasprice,err:=AIMan.GetGasPrice() |

|  |  |
| --- | --- |
| api名称 | SignTxByPrivate |
| api描述 | 根据私钥对交易进行签名 |
| 输入参数 | sendTX \*common.SendTxArgs1, 要签名的交易  from string : 发起交易的from地址  Privatekey \*ecdsa.PrivateKey : from地址使用的私钥  ChainId \*big.Int : 端口指定的ChainId |
| 返回值 | err：错误信息  \*common.SendTxArgs1：签名后的交易 |
| 示例 | trans,err=AIMan.SignTxByPrivate(trans,from,PrivateKey,connection.ChainID) |

|  |  |
| --- | --- |
| api名称 | GetTransactionCount |
| api描述 | 获取当前nonce值 |
| 输入参数 | address：要获取的地址  string："latest"、"earliest"或"pending" |
| 返回值 | \*big.int:指定地址的交易数量 |
| 示例 | nonce, err := AIMan.GetTransactionCount(from, "latest") |

|  |  |
| --- | --- |
| api名称 | GetBalance |
| api描述 | 获取指定区块时给定钱包地址的余额 |
| 输入参数 | string: 要查询余额的地址  Number|String -（可选）如果不设置此值使用man.defaultBlock设定的块，否则使用指定的块。 |
| 返回值 | err：错误信息  []common.RPCBalanceType: 包含给定地址的当前余额的BigNumber实例，单位为wei |
| 示例 | balances,err := AIMan.getBalance("MAN.47kRUvvpaQPHz3faEvesFcLpdYSim") |

|  |  |
| --- | --- |
| api名称 | SendRawTransaction |
| api描述 | 发送一个已经签名的交易。 |
| 输入参数 | type SendTxArgs1 struct {  From string `json:"from"`  To \*string `json:"to"`  Gas \*hexutil.Uint64 `json:"gas"`  GasPrice \*hexutil.Big `json:"gasPrice"`  Value \*hexutil.Big `json:"value"`  Nonce \*hexutil.Uint64 `json:"nonce"`  Data \*hexutil.Bytes `json:"data"`  Input \*hexutil.Bytes `json:"input"`  V \*hexutil.Big `json:"v"`  R \*hexutil.Big `json:"r"`  S \*hexutil.Big `json:"s"`  Currency \*string `json:"currency"`  TxType byte `json:"txType"` //  LockHeight uint64 `json:"lockHeight"` //  IsEntrustTx byte `json:"isEntrustTx"`  CommitTime uint64 `json:"commitTime"`  ExtraTo []\*ExtraTo\_Mx1 `json:"extra\_to"` //  }  Object - 要发送的交易对象。   * from: String - 指定的发送者的地址。如果不指定，使用man.defaultAccount。 * to: String - 交易消息的目标地址，如果是合约创建，则不填. * value: Number|String|BigNumber - 交易携带的货币量，以wei为单位。如果合约创建交易，则为初始的基金。 * gas: Number|String|BigNumber - 交易使用的gas，未使用的gas会退回。 * gasPrice: Number|String|BigNumber - 交易的gas价格 。 * data: String - （可选）或者包含相关数据的字节字符串，如果是合约创建，则是初始化要用到的代码。 * nonce: Number - 整数，使用此值，可以允许你覆盖你自己的相同nonce的，正在pending中的交易。 * V：签名结果 * R：签名结果 * S：签名结果 * Currency：币种名称 * TxType：交易类型（普通转账交易设为0） * LockHeight：保留字段 * IsEntrustTx ：0-自付gas，1-代付gas * CommitTime：提交时间，仅对定时和可撤销交易有效   ExtraTo： 扩展交易（一对多交易填写） |
| 返回值 | String - 32字节的16进制格式的交易哈希串 |
| 示例 | txhash, err := AIMan.SendRawTransaction(raw) |

|  |  |
| --- | --- |
| api名称 | getTransactionReceipt |
| api描述 | 通过一个交易哈希，返回一个交易的收据。  备注：处于pending状态的交易，收据是不可用的。 |
| 输入参数 | String - 交易的哈希  Function - 回调函数，用于支持异步的方式执行[async]。 |
| 返回值 | Object - 交易的收据对象，如果找不到返回null   * blockHash: String - 32字节，这个交易所在区块的哈希。 * blockNumber: Number - 交易所在区块的块号。 * transactionHash: String - 32字节，交易的哈希值。 * transactionIndex: Number - 交易在区块里面的序号，整数。 * from: String -交易发送者的地址。 * to: String -交易接收者的地址。如果是一个合约创建的交易，返回null。 * cumulativeGasUsed: Number - 当前交易执行后累计花费的gas总值。 * gasUsed: Number - 执行当前这个交易单独花费的gas。 * contractAddress: String -创建的合约地址。如果是一个合约创建交易，返回合约地址，其它情况返回null。   logs: Array - 这个交易产生的日志对象数组。 |
| 示例 | var receipt = AIMan.getTransactionReceipt('0x9fc76417374aa880d4449a1f7f31ec597f00b1f6f3dd2d66f4c9c6c445836d8b');  console.log(receipt);  {  "transactionHash": "0x9fc76417374aa880d4449a1f7f31ec597f00b1f6f3dd2d66f4c9c6c445836d8b",  "transactionIndex": 0,  "blockHash": "0xef95f2f1ed3ca60b048b4bf67cde2195961e0bba6f70bcbea9a2c4e133e34b46",  "blockNumber": 3,  "contractAddress": "MAN.38nGzwi5Xn5ApxHXquT8ALaMLpbyG",  "cumulativeGasUsed": 314159,  "gasUsed": 30234,  "logs": [{  // logs as returned by getFilterLogs, etc.  }, ...]  } |

JS接口使用详见： aimanjs接口使用文档说明