# TOT Chain development interface document

目录

[TOT Chain development interface document 1](#_Toc10210)

[一、 Function instructions 3](#_Toc7167)

[二、 Login registration 3](#_Toc12840)

[1． Get encryption public key 3](#_Toc14877)

[2． Check whether the current user name of the whole network exists 4](#_Toc7453)

[3． Check whether the current mailbox of the whole network exists 5](#_Toc13556)

[4． Send verification code 5](#_Toc22170)

[5． Check whether the current registration needs to verify the mailbox 6](#_Toc29465)

[6． Registered account 7](#_Toc8825)

[7． Deployment contract 7](#_Toc10799)

[8． Login 8](#_Toc9279)

[9． Import account private key 9](#_Toc19051)

[10． Export account private key 10](#_Toc769)

[11． Import keystore file 11](#_Toc24414)

[12． Export keystore file 11](#_Toc24986)

[13． Logout 12](#_Toc22491)

[三、 Content management 13](#_Toc30060)

[1． Create type 13](#_Toc30253)

[2． Create Name 13](#_Toc6205)

[3． Del name 14](#_Toc29446)

[4． Create Version 15](#_Toc22002)

[5． Del version 16](#_Toc28869)

[6． Update content 16](#_Toc15673)

[7． Query type 17](#_Toc2564)

[8． Query name 18](#_Toc12894)

[9． Query version 18](#_Toc23954)

[10． Query name list or version list 19](#_Toc15834)

[四、 Transaction 20](#_Toc19271)

[1． Transfer 20](#_Toc23653)

[2． Call contract 21](#_Toc28746)

[3． Query transaction details 21](#_Toc17553)

[4． Query transaction result set 22](#_Toc9926)

[五、 Other 22](#_Toc30262)

[1． Get free gas 22](#_Toc11307)

[2． Buy gas 23](#_Toc18994)

[3． Query the encryption and decryption secret key corresponding to the address 23](#_Toc27227)

[4． Query account address 24](#_Toc11602)

[5． Query address content 24](#_Toc900)

[6． Query block list 25](#_Toc18271)

[7． Blocks 26](#_Toc32106)

[六、 Compatible with Ethereum interface 27](#_Toc6046)

[1． Query block 27](#_Toc4214)

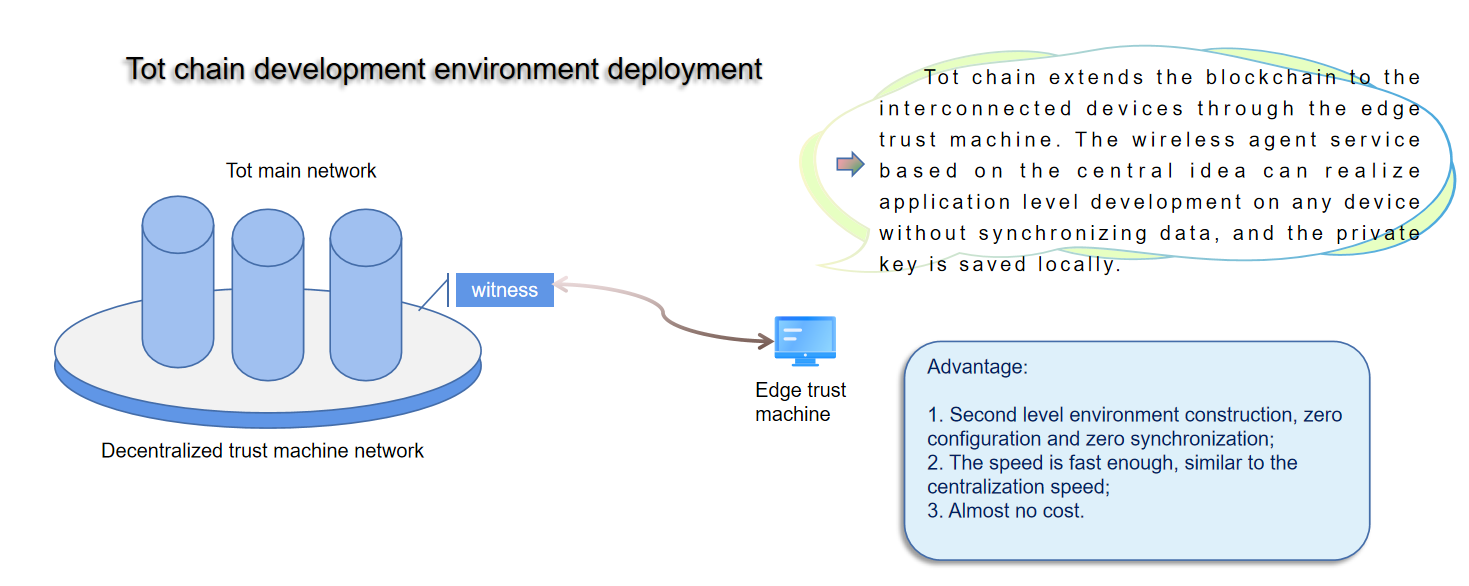
[2． Query transaction 28](#_Toc6724)

[七、 Keyword 29](#_Toc32708)

[八、 Error code 32](#_Toc26294)

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 发布时间 | 内容 | 作者 |
| 1.00 | 2021.10.5 | Basic interface function | ... |
| 1.01 | 2021.10.9 | Adding 5.5 blocks requires adding 6 Ethereum query interfaces | ... |
| 3.01 | 2022.02.16 | All interfaces are compatible and unified, the return value is in JSON format, and the version is upgraded to v3 0 | ... |

1. **Function instructions**



download： http://www.tot.run/download/tmachine.rar

usage：Execute commands under Linux system:

/tmac/tmachine. exe --datadir /tmac\_ data conssole

Note:

1）Execute commands under Linux system: /tmac/tmachine. exe --datadir /tmac\_ data conssole

2) the full execution path must be entered

3) You must specify a data directory for the full path

4) After starting the edge truster, you need to wait for authsuccessful from tot before submitting the blockchain instruction.

1. **Login registration**

Request format description: D\_ Encryption: RSA, pass and none are used for encryption before successful login, and pass is used for encryption after successful login. During the test, you can choose none not to encrypt.

D\_ Session is the callback ID returned from the first connection. All subsequent communications must carry this ID before they can be recognized as the same connection.

In particular:**All d in document\_ The keyword definitions are those that begin with a single capital letter followed by an underscore, and need to be replaced with corresponding definitions, which will not be explained one by one later.**

* 1. **Get encryption public key**

|  |
| --- |
| 请求格式 |
| {  "jsonrpc": "3.0",  "method": "tot\_getKey",  "params": [],  "id": 1  } |
| Return value |
| {      "jsonrpc": "3.0",      "id": 1,      "result": {          "ret": "0",          "err": "",          "content": {              "session": "8f6e7df894f2927e9e747a76",              "pubkey": "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA1ZTs/fVXE8OY5m4qogn3\r\ng93b8Km+i6An/1Mc1KwNv2bszLicS1OAtCwHm68u4eSavsXaMBAkx8PTCPuG1I4X\r\nzHb5TFEZwCNpAlxFgRmnEB+RdFPI8xieHONSGrUni1mTiYfZSBG3rgoI20T5maqp\r\nbEKiBDQmCjEWn2/OFYAvjFpggUCbTPB1WywrH26u9XbFSa+7EIcVPfEnA8IM8ykI\r\nwQYktBx3ew+Z1C1Dsz17MM2Q3WgvhXJdZ2RjI5miPTbyAxFdol3Xk0qd3lJwOk/n\r\nsIM4f4hoNCNmPMVrVqVy4IXFO/dL36DfMimb7cK/2P9sa1JifniFJz0nfwmNaaRz\r\npQIDAQAB\r\n"          }      }  } |
| Return result format: RET return value 0 success other Failure  Err return failure reason  Session callback ID, which should be taken with the subsequent connection  Public key of pubkey service connector |

* 1. **Check whether the current user name of the whole network exists**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_checkAccount",  "params": ["OpCode=OpCode\_Account&SubCode=SubCode\_ExistAccount&D\_Account=tot\_test","D\_Encryption=none"],  "id": D\_Session  } |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_checkAccount",  "params": ["opcode=Account&subcode=Exist&account=wxl\_test","encryp=none"],  "id": "8f6e7df894f2927e9e747a76"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "8f6e7df894f2927e9e747a76",      "result": {          "ret": "0",          "err": "",          "content": {              "address": ""          }      }  }  Address: if it is empty, it means that the account is not used, otherwise the address in use will be returned |

* 1. **Check whether the current mailbox of the whole network exists**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_checkMail",  "params": ["OpCode=OpCode\_Account&SubCode=SubCode\_ExistAccount&D\_Mail=","D\_Encryption=none"],  "id": D\_Session  } |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_checkMail",  "params": ["opcode=Account&subcode=Exist&mail=dd@cn.com","encryp=none"],  "id": "a33fd378ae57294ec72251e7"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "a33fd378ae57294ec72251e7",      "result": {          "ret": "0",          "err": "",          "content": {              "address": ""          }      }  }  Address: if it is empty, it means that the mailbox is not in use, otherwise the address in use will be returned  Err: if an error occurs, an error message is returned |

* 1. **Send verification code**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_sendMail",  "params": ["OpCode=OpCode\_Account&SubCode=SubCode\_SendMail&D\_Mail=","D\_Encryption=none"],  "id": D\_Session  } |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_sendMail",  "params": ["opcode=Account&subcode=SendMail&mail=test@163.com","encryp=none"],  "id": "c1d37567ed3b61f1209d57e9"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "a33fd378ae57294ec72251e7",      "result": {          "ret": "0",          "err": "",          "content": {              "address": ""          }      }  }  RET = 0, sent successfully |

* 1. **Check whether the current registration needs to verify the mailbox**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_isAuthMail",  "params": [],  "id": D\_Session  } |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_isAuthMail",  "params": [],  "id": "c1d37567ed3b61f1209d57e9"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "c1d37567ed3b61f1209d57e9",      "result": {          "ret": "0",          "err": "",          "content": {              "info": "false"          }      }  }  Check whether the current registration needs to be verified. If the mailbox info is false, it does not need to be verified. If it is true, it needs to be verified. In order to prevent the program registration, the system must verify the mailbox for repeated registration of accounts in the same place within 5 minutes, and confirm that only one account can be registered in a valid mailbox. |

* 1. **Registered account**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_register",  "params": ["OpCode=OpCode\_Account&SubCode=OpCode\_Register&D\_Account=dd&D\_Content=d&D\_References = d & D\_Password = 12345678 &D\_Phone = 13816813901 ", "D\_Encryption= "],  "id": D\_Session  } |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_register",  "params":["opcode=Account&subcode=Regsiter&account=wxl\_test&content=d&refer=d &pass=12345678&phone=13816813901&mail=test@protonmail.com&verification=823405", "encryp=none"],  "id": "aa80aa0c6bb0c7d81ac7e5b3"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "b227829636303d832a5222c4",      "result": {          "ret": "0",          "err": "",          "content": {              "address": "0xc3b6472d6370EAf4EB58FFF19Ce1724c9d61892c",              "txhash": "0x0e616dd60d8797a7ede1f2d9a6ef3aff985c9a6a34d35df760642270c526a23c",         }      }  }  Successfully return the transaction hash and the corresponding account address  0xc7fd75384aa827bc03127c7b67c1514de843355a |
|  |

* 1. **Deployment contract**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_sendTransaction",  "params": [{  "from": "13dcb39fc7e7158b9d36c5945e28717872717aa2",  "data": "0x608060405234801561001057600080fd5b50336000806101000a81548173ffffffffffffffffffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffffffffffffffff160217905550611e4f806100606000396000f3fe608060405234801561001057600080fd5b50600436106100b45760003560e01c806385522d841161007157806385522d8419"  },”senderEncryptionMark”, "D\_Encryption=none"],  "id": D\_Session}  //The calling format is compatible with Ethereum |
| Request use case |
| {      "jsonrpc": "3.0",      "method": "tot\_sendTransaction",      "params": [{      "from": "0xc3b6472d6370eaf4eb58fff19ce1724c9d61892c",      "data":"0x608060405234801561001057600080fd5b50336000806101000a81548173ffffffffffffffffffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffffffffffffffff160217905550611e4f806100606000396000f3fe608060405234801561001057600080fd5b50600436106100b45760003560e01c806385522d841161007157806385522d8419"      },"senderEncryptionMark", "encryp=none"],      "id": "2ef7e8faccc82d0eecf262f7"  } |
| Return value |
| {      "jsonrpc": "3.0",      "id": "2ef7e8faccc82d0eecf262f7",      "result": {          "ret": "0",          "err": "",          "content": {              "txhash": "0x7b6553e59d2977e18f5a4e40b4a375af388bde4c19a4f266888c1be1feaecdb7"          }      }  } |

* 1. **Login**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_logon",  "params": ["OpCode=OpCode\_Account&SubCode=SubCode\_Logon&D\_Account=wxl\_test&D\_Password =12345678", "D\_Encryption=rsa"],  "id": D\_Session  } |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_logon",  "params":["opcode=Account&subcode=Logon&account=wxl\_test&pass=12345678", "encryp=none"],  "id": "aa80aa0c6bb0c7d81ac7e5b3"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "fe4a61bf38d1fda89853b985",      "result": {          "ret": "0",          "err": "",          "content": {              "info": "0xc3b6472d6370eaf4eb58fff19ce1724c9d61892c"          }      }  }  After success, return the address corresponding to the account 0xc3b6472d6370eaf4eb58fff19ce1724c9d61892c |

* 1. **Import account private key**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_ImportKey",  "params": ["OpCode=OpCode\_Account&SubCode=SubCode\_Import&D\_Account=wxl\_test&D\_Password =12345678&D\_File1 = dd & D\_File2 = dd ", "D\_Encryption=rsa"],  "id": D\_Session  } |
| Request use case |
| {  "jsonrpc": "2.0",  "method": "tot\_importKey",  "params": ["opcode=Account&subcode=import&account=wxl\_telldd1&pass=12345678&file1=e663a51765134d9484d04c9998cde0ba37c778d2789a5c285c013da1a864739b5d1ce7443c595ea2c375832aea7e298640770e925795bd3e7fa4ca59a2401aa3b35938572d5e280310b440529e5869415aafc0563ac6fffe5d6a96fe8a4dee74a6ed752c6a846474f9d704f74fc07411218fa224599f0dd654d15a16a0e85e220ae484a1b6e9b9d76a964731ac2cace17eda0386e06915e97aff32f9918483472fee44af839c9d1e4b4d0&file2=72607fd237b78417dccccfb4f9079b6ec3c350d8ed76d53b7215ea9602ac4ad2f57a37a1dc3cd0f6ca79f1f913042a7003b447ab0c422d9efd6bcd8d14798ce543a4b37e775500abb2ba465207df819603ab31f5c7ae98ad0665a63910852dcfebd909653c263f0f8fcbb99da5382757d5a81e2e4cf46caf462eed6aafbe3b677f9a1d64fc21d09683e2e9b3eaa3100e2e0d459fa5c76ba537961cfb6ec7fc2a0362087a039575d7785baa65a0ae0e8d", "encryp =none "],  "id": "aa80aa0c6bb0c7d81ac7e5b3"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "130510607259be192697083d",      "result": {          "ret": "0",          "err": "",          "content": {              "info": "0xc3b6472d6370eaf4eb58fff19ce1724c9d61892c"          }      }  } |

* 1. **Export account private key**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_exportKey",  "params":["OpCode=OpCode\_Account&SubCode=SubCode\_Export", "D\_Encryption=rsa"],  "id": D\_Session  } |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_exportKey",  "params": ["opcode=Account&subcode=export", "encryp=none"],  "id": "aa80aa0c6bb0c7d81ac7e5b3"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "b8f9ed77a9c41ea49f191f82",      "result": {          "ret": "0",          "err": "",          "content": {              "file1": "b13958c1163299e51f65dc64537823fecb883ece8bb61060949cb695ff5dfe91a2f22d28e93debf6c5effa49cefa935091ba26f84769ef71f9995226777d22ab8e25ec31728c86b4922f87714a358ba559f94cfcd6ad635f837057524e309ae2d17a4198325e23d9ff5a22bac6934c56183ece2518212fb4ff4fa6a5e877fabfc499656c1bafa9f627ecc7d7d66a79c1331c1b486cdee7ace384560311d45408401bb5cea52d3449c2ea071bb",              "file2": "1a5d74e9e01357ecaeb6c1d20b3e695251baa3f659fed7d309539bd81fcf5ea334c670b7a8275e5b5e63c4c4bf7c0c04d0bc54516cfb84f86420cb5e73947508c8e2ae945d8580c80d08d770ea772bd914ad1af7dcd3933d4cd976ad021b7c7527249a59f41b0e38d2920a5faa1e97e6adb51ce962e0535fccfb6a2dd7d0f00be4c4905f692b259f9f573e6095fb635d6df955edb6270362db271c405c069ea10c9db115ed88192ab5b5bf25a7ed"          }      }  }  After successful return, the two files are extracted through file1 and File2, and there is no order requirement for storage. |

* 1. **Import keystore file**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_importKey",  "params":["OpCode=OpCode\_Account&SubCode=SubCode\_ImportKey&D\_File1=&D\_Account=&D\_Password=", "D\_Encryption=rsa"],  "id": D\_Session  } |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_importKeystore",  "params": ["opcode=Account&subcode=importkey&account=test\_tellffff1&pass=12345678&file1=", "encryp=none"],  "id": "aa80aa0c6bb0c7d81ac7e5b3"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "54c2c71763e4b425f1fc5198",      "result": {          "ret": "0",          "err": "",          "content": {              "address": "0xc3b6472d6370EAf4EB58FFF19Ce1724c9d61892c",              "session": "54c2c71763e4b425f1fc5198",              "pubkey": "",              "content": "0x642dc522abcf67f7439e8329144ebdbe30ab85e2024f6871e8920e8ce7c16973",              "file1": "",              "file2": ""          }      }  } |

* 1. **Export keystore file**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_exportKeystore",  "params":["OpCode=OpCode\_Account&SubCode=SubCode\_EXportKey", "D\_Encryption=rsa"],  "id": D\_Session  } |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_exportKeystore",  "params": ["opcode=Account&subcode=exportkey", "encryp=none"],  "id": "aa80aa0c6bb0c7d81ac7e5b3"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "6084fbc351e03af6909718d8",      "result": {          "ret": "0",          "err": "",          "content": {              "info": ""          }      }  } |

* 1. **Logout**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_logon",  "params":["OpCode=OpCode\_Account&SubCode=SubCode\_Logout", "D\_Encryption=pass"],  "id": D\_Session  } |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_logout",  "params":["opcode=Account&subcode=Logout", "encryp=none"],  "id": "aa80aa0c6bb0c7d81ac7e5b3"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "b8f9ed77a9c41ea49f191f82",      "result": {          "ret": "0",          "err": "",          "content": ""      }  } |

1. **Content management**
   1. **Create type**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_createType",  "params":["OpCode=OpCode\_IOT&SubCode=SubCode\_CreateType&D\_Name=test&D\_Content=", "D\_Encryption=none"],  "id": D\_Session  }  Note: when creating a type, first check the existing type through the query type interface and check whether the current type name is duplicate. If it is duplicate, you will be prompted to prohibit creation. |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_createType",  "params":["opcode=Iot&subcode=createtype&name=soft&content=abcdefg", "encryp=none"],  "id": "291687c0293354a642e3eadf"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "130510607259be192697083d",      "result": {          "ret": "0",          "err": "",          "content": {              "address": "0x6a79f26d03f0fd3c1d65b2ac86d62c2e18a6e32b",              "txhash": "0xf4b56c6de43f0a831fa5533ef33827115526be5e991b401cccfacab65da58fb1"          }      }  }  After successful creation, this type of address and transaction hash |

* 1. **Create Name**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_createType",  "params":["OpCode=OpCode\_IOT&SubCode=SubCode\_CreateName&D\_Address=&D\_Name=&D\_Content=", "D\_Encryption=none"],  "id": D\_Session  }  Note: when creating a name, first check the existing name by querying the name interface under the corresponding type, and check whether the current name is repeated. If it is repeated, you will be prompted to prohibit creation. |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_createName",  "params": ["opcode=Iot&subcode=createname&address=0x6a79f26d03f0fd3c1d65b2ac86d62c2e18a6e32b&name=dev&content=abcdefg", "encryp=none"],  "id": "291687c0293354a642e3eadf"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "291687c0293354a642e3eadf",      "result": {          "ret": "0",          "err": "",          "content": {              "address": "0x03f0eb3e4264733e5bb765970f5e07a36075d2f9",              "txhash": "0xacaa34547259bf272a15b4e814fc019eb304fa9dc6c376523f1d207e8351d75c"          }      }  }  Address and transaction hash of this name after successful creation |

* 1. **Del name**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_delName",  "params":["OpCode=OpCode\_IOT&SubCode=SubCode\_DelName&D\_Address=&D\_Addr1=", "D\_Encryption=none"],  "id": D\_Session  }  D\_ Address = address corresponding to the name type to be deleted  D\_ ADDR1 = name and address to be deleted |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_delName",  "params": ["opcode=Iot&subcode=delName&address=0x6a79f26d03f0fd3c1d65b2ac86d62c2e18a6e32b&addr1=0x03f0eb3e4264733e5bb765970f5e07a36075d2f9&name=dev", "encryp=none"],  "id": "291687c0293354a642e3eadf"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "291687c0293354a642e3eadf",      "result": {          "ret": "0",          "err": "",          "content": {              "address": "0xc3b6472d6370eaf4eb58fff19ce1724c9d61892c",              "txhash": "0xf0a11ca2f689443d27995c3d9be8b9a114b04e665a5a34b7f5d5ac69abbde2b7"          }      }  } |

* 1. **Create Version**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_createVer",  "params":["OpCode=OpCode\_IOT&SubCode=SubCode\_CreateVer&D\_Address=&D\_Addr1=&D\_Name=&D\_Content=", "D\_Encryption=none"],  "id": D\_Session}  Note: when creating a version, first check the existing version by querying the version interface under the corresponding name, and check whether the current version is repeated. If it is repeated, you will be prompted to prohibit creation.  D\_ Address = type address  D\_ ADDR1 = name and address |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_createVer",  "params":["opcode=Iot&subcode=createver&address=0x6a79f26d03f0fd3c1d65b2ac86d62c2e18a6e32b&addr1=0xe0af93f56204461b56deb5e57615f5c705d12fed&name=version&content=abcdefg", "encryp=none"],  "id": "291687c0293354a642e3eadf"} |
| Return value |
| {      "jsonrpc": "2.0",      "id": "291687c0293354a642e3eadf",      "result": {          "ret": "0",          "err": "",          "content": {              "address": "0xc4b1750e4d18496618f3136ffb4d999b56119809",              "txhash": "0xda3c1de88544cecd3b53b17389418c7d8bb05a52f824a05f9089a5048c7bf3d6"          }      }  } |

* 1. **Del version**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_delName",  "params":["OpCode=OpCode\_IOT&SubCode=SubCode\_DelName&D\_Address=&D\_Addr1=&D\_Addr2=", "D\_Encryption=none"],  "id": D\_Session} |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_delVer",  "params": ["opcode=Iot&subcode=delver&address=0xe0af93f56204461b56deb5e57615f5c705d12fed&addr1=0xc4b1750e4d18496618f3136ffb4d999b56119809", "encryp=none"],  "id": "291687c0293354a642e3eadf"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "291687c0293354a642e3eadf",      "result": {          "ret": "0",          "err": "",          "content": {              "address": "0xc3b6472d6370eaf4eb58fff19ce1724c9d61892c",              "txhash": "0xb7b2dd20fe27d0afbd6f53cdeec0346d54ebbdf1610e18940a726433ae63a4fb"          }      }  } |

* 1. **Update content**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_updateContent",  "params":["OpCode=OpCode\_IOT&SubCode=SubCode\_Update&D\_Address=&D\_Name=&D\_Type=&D\_Content=", "encryp=none"],  "id": D\_Session} |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_updateContent",  "params": ["opcode=Iot&subcode=update&address=0x10681bda39977c3E9c8cA2fD750b75C11a0701D9&name=devtest&type=&content=testok", "encryp=none"],  "id": "291687c0293354a642e3eadf"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "291687c0293354a642e3eadf",      "result": {          "ret": "0",          "err": "",          "content": {              "address": "0xc3b6472d6370eaf4eb58fff19ce1724c9d61892c",              "txhash": "0xb1c59accf509d877e24adf366c5a2233d432913b0b568d226cc6ea6d8973c0bb"          }      }  } |

* 1. **Query type**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_queryIotInfo",  "params":["OpCode=OpCode\_IOT&SubCode=SubCode\_QueryIotInfo&D\_Contract=&D\_FunName=D\_Address=", "D\_Encryption=none"],  "id": D\_Session}  D\_ Contract = contract name  D\_ Address = current account address |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_queryIotInfo",  "params": ["opcode=Iot&subcode=QueryIotInfo&Contract=wxl&function=GetType&address=0x37369Ca90E0f1aFC2dC7A1586EBf8e1fE0471e29", "encryp=none"],      "id": "291687c0293354a642e3eadf"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "291687c0293354a642e3eadf",      "result": {          "ret": "0",          "err": "",          "content": {              "info": "249583e47370aa72037f2d3ce410c8de8be6c70d,429c5aa4e577739583657793c3578dbcc1c3d89f,670355fed4a55dd7ca8571d0177c3efb59794490,6a79f26d03f0fd3c1d65b2ac86d62c2e18a6e32b",           }      }  }  Return type address list |

* 1. **Query name**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_queryIotInfo",  "params":["OpCode=OpCode\_IOT&SubCode=SubCode\_QueryIotInfo&D\_Contract=&D\_FunName=&D\_Addr1=", "D\_Encryption=none"],  "id": D\_Session}  D\_ Contract = contract name  D\_ Address = current account address  D\_ ADDR1 = address of corresponding type |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_queryIotInfo",  "params": ["opcode=Iot&subcode=QueryIotInfo&Contract=wxl&function=GetNameFromType&address=0x37369Ca90E0f1aFC2dC7A1586EBf8e1fE0471e29&addr1=0x6a79f26d03f0fd3c1d65b2ac86d62c2e18a6e32b", "encryp=none"],  "id": "291687c0293354a642e3eadf"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "291687c0293354a642e3eadf",      "result": {          "ret": "0",          "err": "",          "content": {              "address": "e0af93f56204461b56deb5e57615f5c705d12fed",          }      }  } |

* 1. **Query version**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_queryIotInfo",  "params":["OpCode=OpCode\_IOT&SubCode=SubCode\_QueryIotInfo&D\_Contract=&D\_FunName=&D\_Addr1=", "D\_Encryption=none"],  "id": D\_Session}  D\_ Contract = contract name  D\_ Address = current account address  D\_ ADDR1 = address of corresponding name |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_queryIotInfo",  "params": ["opcode=Iot&subcode=QueryIotInfo&Contract=wxl&function=GetVerFromName&address=0x37369Ca90E0f1aFC2dC7A1586EBf8e1fE0471e29&addr1=0xd6dc87337bd74727ea34c2e79e4dde2faaf71fc2", "session=7ee6a324fbb3331d62a23e63&encryp=none"],  "id": "291687c0293354a642e3eadf"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "3c707c9238dce859fef9622a",      "result": {          "ret": "0",          "err": "",          "content": {              "info": "null"          }      }  } |

* 1. **Query name list or version list**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_queryIotInfo",  "params":["OpCode=OpCode\_Other&D\_Type=&D\_Address=", "D\_Encryption=none"],  "id": D\_Session}  D\_ Type = have returns the corresponding title and content none returns only the corresponding title |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_queryAddrNameList",  "params": ["opcode=Other&address=0x37369Ca90E0f1aFC2dC7A1586EBf8e1fE0471e29&type=have", "encryp=none"],  "id": "291687c0293354a642e3eadf"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "cb8dfeb15f45f119c4636ce0",      "result": {          "ret": "0",          "err": "",          "content": {              "info": [                  {                      "address": "e0af93f56204461b56deb5e57615f5c705d12fed",                      "name": "dev",                      "time": "1645083594",                      "content": "abcdefg"                  }              ]          }      }  } |

1. **Transaction**
   1. **Transfer**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_sendTransaction",  "params": [{  "from": "13dcb39fc7e7158b9d36c5945e28717872717aa2",  "to": "0x37369Ca90E0f1aFC2dC7A1586EBf8e1fE0471e29",  "value": "0x1"  },”senderEncryptionMark”, "D\_Encryption=none"],  "id": D\_Session}  The calling format is compatible with Ethereum |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_sendTransaction",  "params": [{  "from": "0xc3b6472d6370eaf4eb58fff19ce1724c9d61892c",  "to": "0x6268af5542c4b0ccbb650a0aaccbfcdfbabf6b52",  "value": "0x2E90EDD00"  }, "G0tpOYp4eywxdzsqNMkB85fEK8bFFCRnu4wJ14Wpeq2QrJacqqX/76ty6y0/osUj8foXvMzGVtpbFrjx2BSW8rIzewF89fyh8+evR2ZZqTAYGrYXXrYuB+2+RNUmvyzY", "encryp=pass"],  "id": "54c2c71763e4b425f1fc5198"  } |
| Return value |
| {      "jsonrpc": "3.0",      "id": "2ef7e8faccc82d0eecf262f7",      "result": {          "ret": "0",          "err": "",          "content": {              "txhash": "0x239db776339bef6b24de7a3269d8f76d4cda1c124e8aecc590a93c7d786fc391"          }      }  } |

* 1. **Call contract**

|  |
| --- |
| Refer to the above format and be compatible with Ethereum |
|  |

* 1. **Query transaction details**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_getTransactionReceipt",  "params":["OpCode=OpCode\_Other &D\_Value=", "encryp=rsa"],  "id": D\_Session} |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_getTransactionReceipt",  "params": ["opcode=Other&value=0x7b6553e59d2977e18f5a4e40b4a375af388bde4c19a4f266888c1be1feaecdb7", "encryp=none"],  "id": "08a7d74f27f788b3da1a460d"  } |
| Return value |
| {      "jsonrpc": "3.0",      "id": "08a7d74f27f788b3da1a460d",      "result": {          "BlockHash": "0xcabdb81360c9303ef2ec681893333fea83c1104dc95be5b3807002a24f7a5f0c",          "BlockNumber": 331,          "Txhash": "0x7b6553e59d2977e18f5a4e40b4a375af388bde4c19a4f266888c1be1feaecdb7",          "Txindex": 0,          "From": "0xc62e58e98b2ef6c271568ddef4d80fe3adaa2f1b",          "To": **null**,          "GasUsed": 0,          "ContAddr": "0x27a1d037852fe9cb16cc102130f6cebed7fdf976",          "Root": "asBTcXkKJbmIHtkpuHVl+fNleOJiebOxw5U/J2b4Gho=",          "Status": 0      }  } |

* 1. **Query transaction result set**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_getReceipt",  "params":["OpCode=OpCode\_Other&D\_Session= ", "encryp=rsa"],  "id": D\_Session} |
| Request use case |
| {  "jsonrpc":"3.0",  "method":"tot\_getReceipt",  "params":["opcode=Other&session=08a7d74f27f788b3da1a460d","encryp=none"],  "id": "d73d8d4dd35fb639ac6950e8"} |
| Return value |
| {    "jsonrpc": "3.0",      "id": "08a7d74f27f788b3da1a460d",      "result": {          "ret": "0",          "err": "",          "content": {              "Recips": [                  {                      "Txhash": "0x14d7dd56031b7563e212b4e4f987303239f589aece1c20ef4b15ddc649570092",                      "Receipt": "0x248696d7740eCF289F14a28FfC478E26105fa80F"                  }]}}  }  When the client connects, it will regularly pull the transaction results from the main network to the local, and the local transaction submission thread will regularly detect whether the transaction is successfully linked in the local pool. |

1. **Other**
   1. **Get free gas**

|  |
| --- |
| Request format (you can only get it successfully once a day, the address is empty, and you can get energy for yourself) |
| {  "jsonrpc": "3.0",  "method": "tot\_getFreeGas",  "params":["OpCode=OpCode\_Other&SubCode=SubCode\_GetFreeGas&D\_Address=", "encryp=none"],  "id": D\_Session  } |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_getFreeGas",  "params":["opcode=Other&subcode=getfreegas", "encryp=none"],  "id": "7711b09d8b112406b4492e05"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "7711b09d8b112406b4492e05",      "result": {          "ret": "0",          "err": "",          "content": {              "info": "0x4fbb853c7aaea5c6ee2cfbc545b70b737ffa95e32828c3e3c5751d5e2113e0ea"          }      }  } |

* 1. **Buy gas**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_buyGas",  "params":["OpCode=OpCode\_Other&SubCode=SubCode\_BuyGas&D\_Value=", "encryp=none"],  "id": D\_Session  }  D\_Value=Amount of tokens planned to pay |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_buyGas",  "params":["opcode=Other&subcode=buygas&value=0x1", "encryp=none"],  "id": "7711b09d8b112406b4492e05"} |
| Return value |
|  |

* 1. **Query the encryption and decryption secret key corresponding to the address**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_queryEncryptionKey",  "params":["OpCode=OpCode\_IOT&D\_Address=", "encryp=none"],  "id": D\_Session} |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_queryEncryptionKey",  "params":["opcode=Iot&address=", "encryp=none"],  "id": "f3b9b40ee1cbabcb7943368e"} |
| Return value |
|  |

* 1. **Query account address**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_accountAddress",  "params":["OpCode=OpCode\_Other&SubCode=SubCode\_AccountAddress&D\_Account=", "encryp=none"],  "id": D\_Session} |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_accountAddress",  "params":["opcode=Other&subcode=accountaddr&account=wxl\_telldffff1", "encryp=none"],  "id": "f3b9b40ee1cbabcb7943368e"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "f3b9b40ee1cbabcb7943368e",      "result": {          "ret": "0",          "err": "",          "content": {              "info": "0x4646463139436531373234633964363138393263"          }}  } |

* 1. **Query address content**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_queryAddress",  "params":["OpCode=OpCode\_IOT&SubCode=SubCode\_QueryAddress&D\_Address=&D\_Value=latest", "encryp=none"],  "id": D\_Session} |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_queryAddress",  "params":["opcode=Iot&subcode=queryaddress&address=0xD460DcE7c1364A4b6543CB9D3E999def9b19990F&value=latest", "encryp=none"],  "id": "f3b9b40ee1cbabcb7943368e"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "7711b09d8b112406b4492e05",      "result": {          "ret": "0",          "err": "",          "content": {              "info": {                  "Nonce": 5,                  "Balance": 0,                  "Root": "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",                  "Type": 0,                  "Owner": "0x0000000000000000000000000000000000000000",                  "Grant": "0x0000000000000000000000000000000000000000",                  "GasDate": 0,                  "GasValue": 210000000000000,                  "GasBuyValue": 0,                  "Content": "{\"account\":\"test\_tell80\",\"inBox\":\"0x298a50d95C9d387c1284c496911Cb9daCAe7b8c3\",\"notegp01\":\"0xFe67D19cc000315fEe38fcB6AcD8C72Db6F21a5b\",\"notegp02\":\"0xb0Bf6E7cC3fEcf12bCDf89eA7D43bfCF2dd66c1d\"}"              }          }      }  }  D\_ Addr\_ Balance = "Balance" // assets  D\_ Addr\_ Root = "root" //merkle root of the storage trie  D\_ Addr\_ Owner = "Owner" // Owner  D\_ Addr\_ Grant = "Grant" // Authorized operator  D\_ Addr\_ GasValue = "Gasvalue" / / energy value  D\_ Addr\_ GasBuyValue = "GasBuyValue" // Energy value purchased  D\_ Addr\_ Content = "Content" / / Internet of things device information |

* 1. **Query block list**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_queryBlockList",  "params":["OpCode=OpCode\_Other&SubCode=SubCode\_QueryBlock&D\_nums=5&D\_Number=100", "encryp=none"],  "id": D\_Session} |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_queryBlockList",  "params":["opcode=Other&subcode=queryblock&nums=5&number=100", "session=7ee6a324fbb3331d62a23e63&encryp=none"],  "id": "f3b9b40ee1cbabcb7943368e"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "7711b09d8b112406b4492e05",      "result": {          "ret": "0",          "err": "",          "content": {              "info": "[{\"number\":100,\"tv\":1643012986,\"txs\":1,\"size\":707},{\"number\":99,\"tv\":1643012894,\"txs\":1,\"size\":869},{\"number\":98,\"tv\":1643012834,\"txs\":1,\"size\":683},{\"number\":97,\"tv\":1643012781,\"txs\":1,\"size\":869},{\"number\":96,\"tv\":1643012717,\"txs\":1,\"size\":682}]"          }      }  } |

* 1. **Blocks**

|  |
| --- |
| Request format |
| {  "jsonrpc": "3.0",  "method": "tot\_queryBlockStats",  "params":["OpCode=OpCode\_Other&SubCode=SubCode\_Stats", encryp=none"],  "id": D\_Session} |
| Request use case |
| {  "jsonrpc": "3.0",  "method": "tot\_queryBlockStats",  "params":["opcode=Other&subcode=stats", "encryp=none"],  "id": "f3b9b40ee1cbabcb7943368e"} |
| Return value |
| {      "jsonrpc": "3.0",      "id": "7711b09d8b112406b4492e05",      "result": {          "ret": "0",          "err": "",          "content": {              "info": "{\"total\":323,\"totaltx\":329,\"avegblocktv\":1000,\"txsprsec\":1,\"recentblock\":323,\"RBTime\":1645148725,\"RBTimeLen\":-176095200,\"RTxs\":1,\"txhash\":\"0xa2e797e937d36bc2b7ef175473abe842925750a51836d7d3768c7567c7226ecc\"}"          }      }  }  D\_ Stats\_ Totalnums = "total" // Total number of blocks  D\_ Stats\_ TotalTrans = "totaltx" // Total transactions  D\_ Stats\_ AvegBlockTime = "avegblocktv" // Average block out time  D\_ Stats\_ TxsPrSec = "txsprsec" // Transaction Per Second  D\_ Stats\_ RecentBlock = "recentblock" // Nearest block  D\_ Stats\_ RecentTsx = "txhash" // Recent transactions |
|  |

1. **Compatible with Ethereum interface**

|  |
| --- |
| **Tot chain is compatible with Ethereum interface. The main matters of calling method are as follows:**  **1. Version is defined as 3.0**  **2. ID is the session ID corresponding to the current user's login**  **3.Gas, gasplice background ignored** |

* 1. **Query block**

|  |
| --- |
| Request format |
| {"jsonrpc":"3.0","method":"eth\_getBlockByNumber","params":["0x1", **true**],  "id": "f3b9b40ee1cbabcb7943368e"} |
| Return value |
| {      "jsonrpc": "2.0",      "id": 1,      "result": {          "difficulty": "0x20000",          "extraData": "0xda8301091a846765746888676f312e31332e358777696e646f7773",          "gasLimit": "0xffc00001",          "gasUsed": "0x0",          "hash": "0xdfedee9f480c4a2e0fed5b79d56b655e88ecc5c867b86c96e8cfc90b7ac2ce0f",          "logsBloom": "0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",          "miner": "0x0000000000000000000000000000000000000000",          "mixHash": "0x0000000000000000000000000000000000000000000000000000000000000000",          "nonce": "0x0000000000000000",          "number": "0x1",          "parentHash": "0x587553e293b90da898badfc54600eb6e9304211d236cb9d50112e9e2b798acc3",          "receiptsRoot": "0x9b7c4ce7e2103ddedde0aec3d579868158436a13233d040094c635f7692bf705",          "sha3Uncles": "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",          "size": "0x2e4",          "stateRoot": "0xbeeb30d00469b1bf14fe032c163ae4afcfa348ac59e35a9819344657c2f206a7",          "timestamp": "0x615e8a65",          "totalDifficulty": "0x0",          "transactions": [              {                  "blockHash": "0xdfedee9f480c4a2e0fed5b79d56b655e88ecc5c867b86c96e8cfc90b7ac2ce0f",                  "blockNumber": "0x1",                  "from": "0x37369ca90e0f1afc2dc7a1586ebf8e1fe0471e29",                  "gas": "0x0",                  "gasPrice": "0x0",                  "hash": "0x9214926e347e85b13dab5cb7dff2f24c26555c69bbc8fdac337d0c10ddfcaae5",                  "input": "0x01fe01fe01fe01fe6f70636f64653d4f7468657226737562636f64653d6765746672656567617326616464726573733d30783337333639436139304530663161464332644337413135383645426638653166453034373165323926646174653d3230323131303037",                  "nonce": "0x0",                  "to": "0x37369ca90e0f1afc2dc7a1586ebf8e1fe0471e29",                  "transactionIndex": "0x0",                  "value": "0x0",                  "v": "0x1c",                  "r": "0xa14192857e44ad79cb4237dbb0728df9885aa08d2c01870b6369c951c7db5177",                  "s": "0x1d1c45f0fd26294dfc835288c7541081e4b6a17e91a26af28c1ddd6aab7e3642"              }          ],          "transactionsRoot": "0x3a82b5c6d58dc830b4091d2468d0fd275d9c4c8da67620b763a0c655c8da06b0",          "uncles": []      }  } |

* 1. **Query transaction**

|  |
| --- |
| Request format |
| {"jsonrpc":"3.0","method":"eth\_getTransactionReceipt","params":["0x9214926e347e85b13dab5cb7dff2f24c26555c69bbc8fdac337d0c10ddfcaae5"],  "id": "f3b9b40ee1cbabcb7943368e"} |
| Return value |
| {      "jsonrpc": "2.0",      "id": 1,      "result": {          "blockHash": "0xdfedee9f480c4a2e0fed5b79d56b655e88ecc5c867b86c96e8cfc90b7ac2ce0f",          "blockNumber": "0x1",          "contractAddress": **null**,          "cumulativeGasUsed": "0x0",          "from": "0x37369ca90e0f1afc2dc7a1586ebf8e1fe0471e29",          "gasUsed": "0x0",          "logs": [],          "logsBloom": "0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",          "root": "0xb01b7d8ff4b772efbf66a0fd500c6481ab3000cb5b4f945529ea1802dc221d28",          "to": "0x37369ca90e0f1afc2dc7a1586ebf8e1fe0471e29",          "transactionHash": "0x9214926e347e85b13dab5cb7dff2f24c26555c69bbc8fdac337d0c10ddfcaae5",          "transactionIndex": "0x0"      }  } |

1. **Keyword**

|  |  |  |
| --- | --- | --- |
| **name** | **code** | **desc** |
| Sys\_IotWxl | iotwxl |  |
| OpCode | opcode | Main operation code |
| SubCode | subcode | Sub operation code |
| OpCode\_Account | Account | Account operation |
| OpCode\_IOT | Iot | Internet of things |
| OpCode\_Other | Other | other |
| SubCode\_ExistAccount | Exist | Does the account exist |
| SubCode\_AddAccount | Add | Add account |
| SubCode\_Changepass | changepass | Change Password |
| SubCode\_Logon | Logon | Sign in |
| SubCode\_Logout | Logout | Log out |
| SubCode\_GetAccount | GetAccount | Get account |
| SubCode\_ListAccount | ListAccount | Get account list |
| SubCode\_DelAccount | DelAccount |  |
| SubCode\_Import | import |  |
| SubCode\_Export | export | export |
| OpCode\_Product | Product | Product operation |
| SubCode\_Show | Show | View products |
| SubCode\_ShowProduct | ShowPData | View version product content |
| SubCode\_ShowPVER | ShowPvER | View content |
| SubCode\_AddProduct | AddPData | Add products |
| SubCode\_DelProduct | DelPData | Delete product |
| SubCode\_DelPVer | DelPVer | Delete product version |
| OpCode\_Register | Regsiter | Registration operation |
| SubCode\_GetNonce | GetNonce | Get registration code |
| SubCode\_AuthNonce | AuthNonce | Certification registration code |
| SubCode\_GetFreeGas | getfreegas | Get free energy |
| SubCode\_CreateType | createtype | build type |
| SubCode\_CreateName | createname | create name |
| SubCode\_CreateVer | createver | Create version |
| SubCode\_QueryIotInfo | QueryIotInfo | Query IOT information |
| SubCode\_DelType | deltype | delete type |
| SubCode\_DelName | delName | Delete name |
| SubCode\_DelVer | delVer | Delete version |
| SubCode\_Update | update | Content update |
| SubCode\_QueryAddress | queryaddress | Query address |
| SubCode\_BuyGas | buygas | Buy energy |
| SubCode\_AccountAddress | accountaddr | Account address |
| SubCode\_QueryBlock | queryblock | Query block information |
| OpCode\_Command | command | Command operation |
| SubCode\_Exit | exit | Exit the system |
| SubCode\_Check | check | Check operation status |
| D\_Username | username |  |
| D\_Account | account | Account address |
| D\_Content | content | Remarks description information |
| D\_Address | address | Account address |
| D\_References | refer | Recommender |
| D\_OwnUser | ownuser | owner |
| D\_Password | pass | password |
| D\_Phone | phone | Bind phone number |
| D\_Type | type | type |
| D\_Nmae | name | name |
| D\_Veriion | version | edition |
| D\_PasswordEn | passen |  |
| D\_CreateAddress | caddress |  |
| D\_Private | private |  |
| D\_Subject | subject |  |
| D\_Name | name |  |
| D\_Addr1 | addr1 |  |
| D\_Addr2 | addr2 |  |
| D\_Productname | pname |  |
| D\_Productdata | Product |  |
| D\_PVername | Pvername |  |
| D\_PVerdata | Pver |  |
| D\_EncodeI | enodeid |  |
| D\_Nonce | nonce |  |
| D\_Ret | ret |  |
| D\_Successful | 0 |  |
| D\_Fail | -1 |  |
| D\_nums | nums |  |
| D\_Status | status |  |
| D\_Err | err |  |
| D\_Session | session | Session ID |
| D\_Date | date | date |
| D\_GetGas | getgas | Get free energy every day |
| D\_BuyGas | buygas | Buy energy |
| D\_Register | register | register |
| D\_Number | number | number |
| S\_FindTeamer | findTeamer |  |
| S\_Query | queryRec |  |
| S\_Modify | updateRec |  |
| S\_AyncUpdate | Syncupate |  |
| S\_AdjuestFollow | adfollow |  |
| S\_Sync | download |  |
| S\_Downloader | downloader |  |
| C\_DownLoad | download |  |
| OpCode\_Put | putkey | Save value |
| OpCode\_Get | getkey | Value |
| OpCode\_QueryList | querylis | query |
| D\_Token | token |  |
| D\_Key | key |  |
| D\_Value | value |  |
| D\_QueryPrefix | qprifix |  |
| D\_QueryRetId | qretid |  |
| D\_QueryMaxnums | maxnums |  |
| D\_Contract | Contract |  |
| D\_FunName | function |  |
| D\_Price | price |  |
| D\_PublicKey | pubkey |  |
| D\_Sign | sign |  |
| D\_Encryption | encryp | encryption |
| D\_File1 | file1 |  |
| D\_File2 | file2 |  |
| D\_Addr\_Balance | Balance | assets |
| D\_Addr\_Root | root | merkle root of the storage trie |
| D\_Addr\_Owner | Owner" | Owner |
| D\_Addr\_Grant | Grant | Authorized operator |
| D\_Addr\_GasValue | GasValue | Energy value |
| D\_Addr\_GasBuyValue | GasBuyValue | Energy value purchased |
| D\_Addr\_Content | Content | Internet of things device information |
| D\_Stats\_Totalnums | total | Total number of blocks |
| D\_Stats\_TotalTrans | totaltx | Total transactions |
| D\_Stats\_AvegBlockTime | avegblocktv | Average block out time |
| D\_Stats\_TxsPrSec | txsprsec | Transaction Per Second |
| D\_Stats\_RecentBlock | recentblock | Nearest block |
| D\_Stats\_RecentTsx | txhash | Recent transactions |

1. **Error code**

|  |  |  |
| --- | --- | --- |
| **Code** | **Content** | **desc** |
| 1000 | Unrecognized error |  |
| 1001 | Timeout | request timeout |
| 1002 | Param format fail | Request parameter format error |
| 1003 | Session not found! | The reply was not found |
| 1004 | Function Disabled | Disabled functions |
| 1005 | The user has not logged in yet ！ | Not logged in yet |
| 1006 | Parameter format error | Request parameter format error |
| 1007 | The data is too long to exceed 1024 |  |
| 1008 | Error parsing encrypted file ! |  |
| 1009 | "Session lost! |  |
| 1010 | Decrypt error! |  |
| 1011 | Find account addr error! |  |
| 1012 | The trust has not been registered |  |
| 1013 | Find Rec Fail! |  |
| 1014 | Rec count is 0 |  |
| 1015 | Json parse error |  |