BHP 算力公链对接指南

V1.0rc

BHP 算力公链团队

2018-09

目 录

1 简介

本文档以交易所与 BHP 算力公链对接为例。

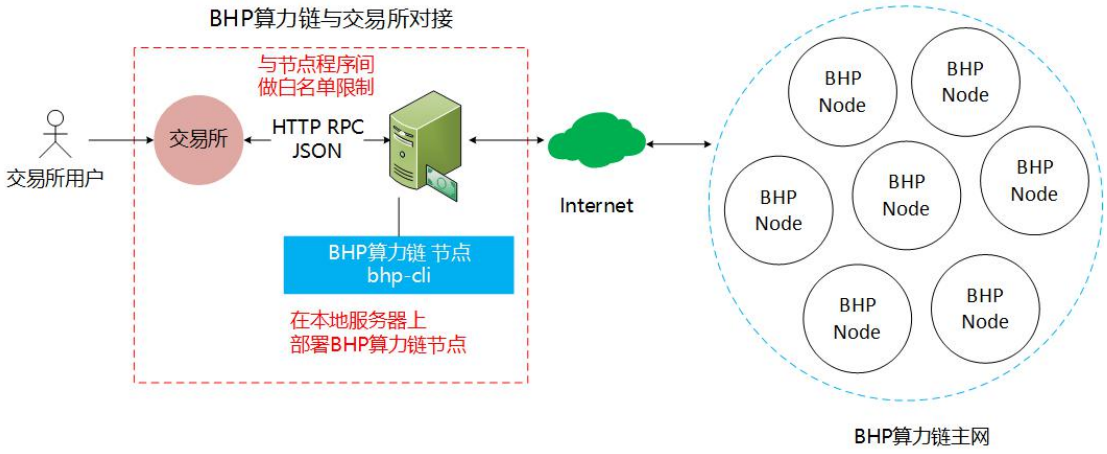
BHP 算力公链中的全局资产是 BHP，使用 UTXO 模型来管理资产。交易所主要处理 BHP 的充币、提币等操作。交易所需要在本地部署 BHP 算力公链的钱包节点：bhp-cli。在 BHP 算力公链的网络中充当一个普通节点，同时该程序也是一个跨平台的钱包，处理各种资产的相关交易。综上，交易所对接需要完成以下操作：

- 在本地服务器中部署 bhp-cli 节点
- 使用 bhp-cli 客户端处理全局资产交易

BHP 全局资产：0x13f76fabfe19f3ec7fd54d63179a156bafc44afc53a7f07a7a15f6724c0aa854

2 部署钱包节点

交易所需要在本地服务器上部署钱包节点：bhp-cli，交易所必须使用白名单或防火墙以屏蔽外部服务器请求，否则会有重大安全隐患。交易所与 BHP 算力公链对接拓扑图：



1、安装运行环境：首先安装 .net core 环境：

- Linux (ubuntu 16.04/ubuntu 17.10)，安装 .NET Core Runtime。
- Windows 7/Windows 10/Windows server，安装 .NET Core 和 .NET Framework。

2、安装钱包程序

(1) 在 Github 上下载节点程序。直接解压至本地目录即可。

(2) 对于 Linux 系统，需要安装 LevelDB 和 SQLite3 开发包。例如，在 ubuntu 17.10 上输入以下命令：

```
sudo apt-get install libleveldb-dev sqlite3 -dev libunwind8-dev
```

(3) 对于 Windows 系统，bhp-cli 的安装包中已经包含了 LevelDB，可跳过第 2 步。

(4) Windows 系统可使用 GUI 钱包程序，进行手工转账。

3、启动钱包节点

打开命令行，定位到 bhp-cli 所在目录，输入以下命令启动节点：

```
dotnet bhp-cli.dll
```

启动钱包节点时，可以带以下参数：

序列	步骤	输入命令
1	运行客户端	dotnet bhp-cli.dll
2	打开 rpc 接口	--rpc
3	直接连接种子节点	--nopeers

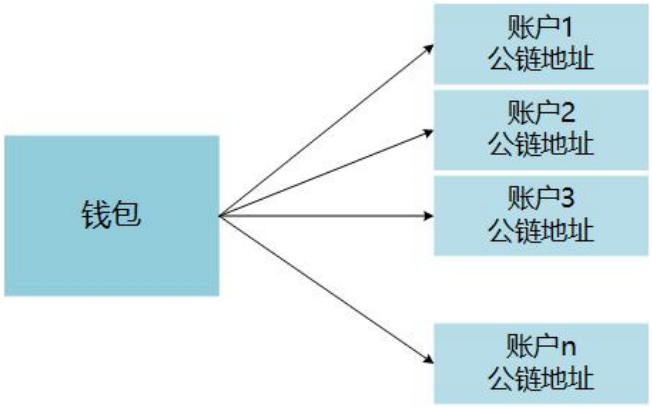
如果想启动节点的同时开启 API，外部程序可通过此 API 接口直接如向指定地址转账，获得指定高度的区块信息，获得指定的交易等。

```
dotnet bhp-cli.dll --rpc
```

3 创建钱包

交易所只需创建一个钱包，在钱包中可创建多个用户地址。

交易所需要创建一个在线钱包管理用户充币地址。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，一定要保管好钱包文件和钱包密码，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，通常一个钱包文件可以存储用户所有充币地址。也可以使用一个冷钱包（离线钱包）作为更安全的存储方式。



★bhp-cli 钱包支持两种格式的钱包，一种是一直使用的 sqlite 钱包（格式为.db3），另一种是新支持的 BRC6 标准的钱包（格式为.json）。建议交易所使用 sqlite 钱包。

请按照以下步骤创建钱包：

1、在控制台输入命令 ：

```
bhp>create wallet <path>
```

其中 <path> 为钱包路径及名称，扩展名根据所使用的钱包种类来设定，可以是 .db3 也可以是 .json（如无扩展名，则钱包格式为 BRC6 钱包）。如 create wallet /home/mywallet.db3。

2、设置钱包密码。

4 生成充币地址

一个钱包可以存储多个地址，交易所需要为每个用户生成一个充币地址。充币地址有两种生成方式：

1、用户第一次充币（BHP）时，程序动态创建 BHP 地址，优点：无需人工定期创建地址；缺点：不方便备份钱包。要动态创建地址，可以使用 bhp-cli 的 API 中的 getnewaddress 方法实现。程序会返回创建的地址。

请求正文：

```
{
  "jsonrpc": "2.0",
  "method": "getnewaddress",
  "params": [],
  "id": 1
}
```

响应正文：

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": "AVHcdW3FGKbPWGHNhkPjgVgi4GGndiCxdo"
}
```

响应说明：返回新创建的地址

2、交易所提前创建一批 BHP 地址，并在用户第一次充币（BHP）时，给用户分配一个 BHP 地址。优点：方便备份钱包；缺点：当地址不足时需要人工创建 BHP 地址。要批量创建地址，执行 bhp-cli 的 create address [n]命令，地址会自动导出到 address.txt 文件。方括号为可选参数，默认值为 1。例如要一次创建 100 个地址，在命令行输入：

```
bhp>create address 100
```

★无论采用哪种方式，交易所需要将生成的地址导入到数据库中，作为充币地址分配给用户。

一般建议交易所采用第二种方式，这样可以减少外界对钱包的操作，有利于钱包的稳定运行。

5 对接程序

对于全局资产 BHP，交易所需要进行以下功能的开发：

1、使用 bhp-cli 的 API 接口 （getblock 方法）监控新区块。

2、根据交易信息完成用户充币。

3、存储交易所相关交易记录。

6 用户充币

关于用户充币，交易所需要了解以下内容：

1、BHP 算力公链只有一条主链，没有侧链，不会分叉，也不会有孤立区块。

2、所有记录在 BHP 算力公链中的交易都是不可篡改的，即一个确认就代表充币成功。

3、一般来讲，交易所充币地址里的余额并不等于用户在交易所里的余额，有以下原因：

（1）在转账或提币时，BHP 钱包会从一个或多个地址中找到即能满足需求又使用总输入最小的零钱作为本次交易的输入，而不会将指定地址的零钱作为交易输入（除非交易所重写了 bhp 钱包的部分代码使其满足自身需求）。

（2）其他操作，例如交易所将一部分资产转移到交易所的冷钱包等。

4、BHP 算力公链钱包是一个全节点，要保持在线才能同步区块，可以通过 bhp-cli 的 show state 命令查看区块同步状态，在命令行输入：

```
bhp>show state

block: 99/99/99, connected: 10
```

含义为：钱包高度 99/区块高度 99/区块头高度 99，连接节点数为 10.

★假设该节点与 P2P 网络充分连接，当区块高度=区块头高度时，代表节点同步完成。当钱

包高度=区块高度=区块头高度时，代表节点同步完成且钱包索引建立完成。

5、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，只有充币提币才上链。

7 充币记录

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币提币交易。如果有充币交易就要修改数据库中的用户余额。

bhp-cli API 中的 getblock [verbose] 方法提供了获取区块信息的功能，该方法中的 为区块索引。[verbose] 默认值为 0，表示返回的是区块序列化后的信息，用 16 进制字符串表示，如果从中获取详细信息需要反序列化。[verbose] 为 1 时返回的是对应区块的详细信息，用 Json 格式字符串表示。更多信息请参阅 getblock 方法。

示例 1 返回区块序列化后的信息：

请求正文（根据区块散列获取）：

```
{
  "jsonrpc": "2.0",
  "method": "getblock",
  "params": ["773dd2dae4a9c9275290f89b56e67d7363ea4826dfd4fc13cc01cf73a44b0d0e"],
  "id": 1
}
```

请求正文（根据区块索引）：

```
{
  "jsonrpc": "2.0",
  "method": "getblock",
  "params": [1],
  "id": 1
}
```

响应正文：

```
{
  "jsonrpc": "2.0",
  "id": 1,
```

```
  "result":
"000000002deadfa82cbc4682f5800ec72a8d8bd6afa469af5b2de83a51d28795a893222816f8081
bf1054136cca420f807f844a958b2dea482dfc99d2538ef9c77d13320f9263659d4220f00878f40bd
841c552a59e75d652b5d3827bf04c165bbe9ef95cca4bf5501fd450140b514d8562ad3badac0e097
a502a43c58e23c75029dad8ccdb3b1ce221067d73d5612950e38c7565d6b166ef62894399a6f152
c38a1bdb8c7d3715f75f20c1c7340e443f55108c5eefd99f954e06b21e97a4f0cf64dbd4e52426c27f
7046cd880d6a7b1a507131c39afa48b9cac16411d6f84ec2f0b5d9977e5f1e3ce760a127b31409b8
a52714b37a3b0970a19b4fb2669d2aa41ea85e05e68dfb03a197d505282dd53846ca58b1457504c
65759a9ceb8f84f5148dec71727e9c743e986092728174401862c08611338be8e352b9110b2bb6d
11ce0485286d857162deb417f1cb920d6727f8e6edbe1b7fce8d9b122523d5b45cfd02ab1ca002a5
8e28c8903ad764a84409dfcbda69cef1164936212e8e5d91965c8a976dc8dbcb5ea7d2f2d2f0105d
adb902924559fede016a1f76a2c7ab0ff89a6446b0c19c88375906c8b9eccb61bc1f1552102486fd1
5702c4490a26703112a5cc1d0923fd697a33406bd5a1c00e0013b09a7021024c7b7fb6c310fccf1ba
33b082519d82964ea93868d676662d4a59ad548df0e7d2102aaec38470f6aad0042c6e877cfd808
7d2676b0f516fddd362801b9bd3936399e2103b209fd4f53a7170ea4444e0cb0a6bb6a53c2bd016
926989cf85f9b0fba17a70c2103b8d9d5771d8f513aa0869b9cc8d50986403b78c6da36890638c3d
46a5adce04a2102ca0e27697b9c248f6f16e085fd0061e26f44da85b58ee835c110caa5ec3ba55421
02df48f60e8f3e01c48ff40b9b7f1310d7a8b2a193188befe1c2e3df740e89509357ae010000878f40
bd00000000"
}
```


示例 2 返回 JSON 格式：

请求正文：


```
    {
      "jsonrpc": "2.0",
      "method": "getblock",
      "params":
["773dd2dae4a9c9275290f89b56e67d7363ea4826dfd4fc13cc01cf73a44b0d0e", 1],
      "id": 1
    }
```


响应正文：


```
    {
      "jsonrpc": "2.0",
      "id": 1,
      "result": {
        "hash":
"0x773dd2dae4a9c9275290f89b56e67d7363ea4826dfd4fc13cc01cf73a44b0d0e",
        "size": 686,
        "version": 0,
        "previousblockhash":
"0x282293a89587d2513ae82d5baf69a4afd68b8d2ac70e80f58246bc2ca8dfea2d",
        "merkleroot":
```

```
"0xce52b4ed7fc69fe044d355bedcf55e070f5b8e661e3e4380cd513aef6224de42",
            "time": 1496721145,
            "index": 991956,
            "nonce": "2a551c84bd408f87",
            "nextconsensus": "APyEx5f4Zm4oCHwFWiSTaph1fPBxZacYVR",
            "script": {
               "invocation":
"40b514d8562ad3badac0e097a502a43c58e23c75029dad8ccdb3b1ce221067d73d5612950e38c7
565d6b166ef62894399a6f152c38a1bdb8c7d3715f75f20c1c7340e443f55108c5eefd99f954e06b2
1e97a4f0cf64dbd4e52426c27f7046cd880d6a7b1a507131c39afa48b9cac16411d6f84ec2f0b5d99
77e5f1e3ce760a127b31409b8a52714b37a3b0970a19b4fb2669d2aa41ea85e05e68dfb03a197d5
05282dd53846ca58b1457504c65759a9ceb8f84f5148dec71727e9c743e986092728174401862c0
8611338be8e352b9110b2bb6d11ce0485286d857162deb417f1cb920d6727f8e6edbe1b7fce8d9b
122523d5b45cfd02ab1ca002a58e28c8903ad764a84409dfcbda69cef1164936212e8e5d91965c8a
976dc8dbcb5ea7d2f2d2f0105dadb902924559fede016a1f76a2c7ab0ff89a6446b0c19c88375906c
8b9eccb61bc1",
               "verification":
"552102486fd15702c4490a26703112a5cc1d0923fd697a33406bd5a1c00e0013b09a7021024c7b7
fb6c310fccf1ba33b082519d82964ea93868d676662d4a59ad548df0e7d2102aaec38470f6aad004
2c6e877cfd8087d2676b0f516fddd362801b9bd3936399e2103b209fd4f53a7170ea4444e0cb0a6b
b6a53c2bd016926989cf85f9b0fba17a70c2103b8d9d5771d8f513aa0869b9cc8d50986403b78c6d
a36890638c3d46a5adce04a2102ca0e27697b9c248f6f16e085fd0061e26f44da85b58ee835c110c
aa5ec3ba5542102df48f60e8f3e01c48ff40b9b7f1310d7a8b2a193188befe1c2e3df740e89509357
ae"
            },
            "tx": [
               {
                  "txid":
"0x2033d1779cef38259dc9df82a4deb258a944f807f820a4cc364105f11b08f816",
                  "size": 10,
                  "type": "MinerTransaction",
                  "version": 0,
                  "attributes": [],
                  "vin": [],
                  "vout": [],
                  "sys_fee": "0",
                  "net_fee": "0",
                  "scripts": [],
                  "nonce": 3175124871
               }
            ],
            "confirmations": 20,
            "nextblockhash":
"0x0b08e2eeed05c70f27293521c47f7f60dfc29f9f299ae9909a8552a4a87db7a2"
```

```
    }
}
```

获取的区块信息中包含了交易输入和交易输出，交易所需要记录下所有和自己相关的交易，作为用户充币提币的交易记录。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币地址对应的用户 BHP 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币记录，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。

★说明：

（1）getblockcount 返回的是主链中的区块数量，getblock 第一个参数是区块索引，区块索引 = 区块高度 = 区块数量 - 1，所以如果 getblockcount 返回 1234，调用 getblock 1234 将获取不到结果，而应该调用 getblock 1233。

请求正文：

```
{
  "jsonrpc": "2.0",
  "method": "getblockcount",
  "params": [],
  "id": 1
}
```

响应正文：

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": 1233
}
```

（2）交易所充币提币交易的交易类型都是 ContractTransaction，交易所在遍历区块中的所有交易时，只需关心 ContractTransaction。

（3）每个区块的第一个交易必定是 MinerTransaction，在遍历交易时可以忽略或跳过。

（4）BHP 系统中的一切事务都以交易为单位进行记录。

8 用户提币

关于用户提币，交易所需要完成以下操作：

1、在 bhp-cli 中，执行 open wallet <path> 命令打开钱包：

```
bhp>open wallet <path>
```

2、记录用户提币，修改用户账户余额。

3、（可选）客服处理提币申请。

4、使用 bhp-cli 的 API 接口 sendtoaddress <asset_id> <address> <value> 方法 ，向用户提币地址发送交易。更多信息，请参阅 sendtoaddress 方法。

- <asset_id> ： 资产 ID
- <address> ： 提币地址
- <value> ： 提币金额

要向多个地址批量发送交易，可以使用 API sendmany 方法。

```
sendmany 参数说明：
    <outputs_array> [fee=0] [change_address]
    outputs_array：数组，数组中的每个元素的数据结构如下：
    {"asset": \<asset>,"value": \<value>,"address": \<address>}
    asset：资产 ID（资产标识符），即该资产在注册时的 RegistTransaction 的交易 ID。其余资产 ID 可以通过 [CLI 命令](../../cli.md) 中的 `list asset` 命令查询，也可以在区块链浏览器中查询
```

value：转账金额

address：收款地址

fee：手续费，可选参数，默认为 0

change_address：找零地址，可选参数，默认为钱包中第一个标准地址

调用示例

请求正文：

```
{
    "jsonrpc": "2.0",
    "method": "sendmany",
    "params": [
        [
            {
                "asset":
"0x13f76fabfe19f3ec7fd54d63179a156bafc44afc53a7f07a7a15f6724c0aa854",
                "value": 1,
                "address": "AbRTHXb9zqdqn5sVh4EYpQHGZ536FgwCx2"
            },
            {
                "asset":
"0x13f76fabfe19f3ec7fd54d63179a156bafc44afc53a7f07a7a15f6724c0aa854",
                "value": 1,
                "address": "AbRTHXb9zqdqn5sVh4EYpQHGZ536FgwCx2"
            }
        ]
    ],
    "id": 1
}
```

请求正文（包含手续费和找零地址）：

```
{
    "jsonrpc": "2.0",
    "method": "sendmany",
    "params": [
        [
            {
                "asset":
"0x13f76fabfe19f3ec7fd54d63179a156bafc44afc53a7f07a7a15f6724c0aa854",
                "value": 1,
```

```
                        "address": "AbRTHXb9zqdqn5sVh4EYpQHGZ536FgwCx2"
                    },
                    {
                        "asset":
"0x13f76fabfe19f3ec7fd54d63179a156bafc44afc53a7f07a7a15f6724c0aa854",
                        "value": 1,
                        "address": "AbRTHXb9zqdqn5sVh4EYpQHGZ536FgwCx2"
                    }
                ],
                0,
                "AbRTHXb9zqdqn5sVh4EYpQHGZ536FgwCx2"
            ],
            "id": 1
        }
```


响应正文：


```
    {
        "jsonrpc": "2.0",
        "id": 1,
        "result": {
            "txid":
"0x55ba819b50f5821298328f3bf9bb17e088afc900cf2ad7dbfc03d49940b5cf30",
            "size": 322,
            "type": "ContractTransaction",
            "version": 0,
            "attributes": [],
            "vin": [
                {
                    "txid":
"0x06de043b9b914f04633c580ab02d89ba55556f775118a292adb6803208857c91",
                    "vout": 1
                }
            ],
            "vout": [
                {
                    "n": 0,
                    "asset":
"0x13f76fabfe19f3ec7fd54d63179a156bafc44afc53a7f07a7a15f6724c0aa854",
                    "value": "1",
                    "address": "AbRTHXb9zqdqn5sVh4EYpQHGZ536FgwCx2"
                },
                {
                    "n": 1,
                    "asset":
```

```
"0x13f76fabfe19f3ec7fd54d63179a156bafc44afc53a7f07a7a15f6724c0aa854",
                        "value": "1",
                        "address": "AbRTHXb9zqdqn5sVh4EYpQHGZ536FgwCx2"
                    },
                    {
                        "n": 2,
                        "asset":
"0x13f76fabfe19f3ec7fd54d63179a156bafc44afc53a7f07a7a15f6724c0aa854",
                        "value": "495",
                        "address": "AK5q8peiC4QKwuZHWX5Dkqhmar1TAGvZBS"
                    }
                ],
                "sys_fee": "0",
                "net_fee": "0",
                "scripts": [
                    {
                        "invocation":
"406e545e30a6b39f71a7a40f1d4937939b9e1ca38851449842a2e2318bd499afd9c89f0c96658923e3e435ee91192e9dbf101d81a240fa7c953ac0c322d2f2b980",
                        "verification":
"2103cf5ba6a9135f8eaeda771658564a855c1328af6b6808635496a4f51e3d29ac3eac"
                    }
                ]
            }
        }
```


响应说明：

1) 返回如上的交易详情说明交易发送成功，否则交易发送失败

2) JSON 格式不正确，会返回 Parse error

3) 如果签名不完整会返回待签名的交易

4) 如果余额不足会返回错误信息

5、从返回的 Json 格式交易详情中提取交易 ID，记录在数据库中。

6、等待区块链确认，确认后将提币记录标志为提币成功。

类似充币时对区块链的监控，提币也一样，监控时若发现区块中的某个交易 ID 与提币记录中的交易 ID 相等，则该交易已经确认，即提币成功。

★说明：

1、提币金额为实际金额，并非乘以 10^8 后的金额。

2、BHP 转账金额可以是小数。

9 查询余额

根据账户地址，查询账户资产信息可以使用 API getaccountstate 方法。参数 address 指以 A 开头的 34 位长度的账户地址。

请求正文：

```
{
  "jsonrpc": "2.0",
  "method": "getaccountstate",
  "params": ["AJBENSwajTzQtwyJFkiJSv7MAaaMc7DsRz"],
  "id": 1
}
```

响应正文：

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": {
        "version": 0,
        "script_hash": "0x1179716da2e9523d153a35fb3ad10c561b1e5b1a",
        "frozen": false,
        "votes": [],
        "balances": [
            {
                "asset":
"0x13f76fabfe19f3ec7fd54d63179a156bafc44afc53a7f07a7a15f6724c0aa854",
                "value": "94"
            }
        ]
    }
}
```

响应说明：

1) script_hash：合约脚本散列，在 BHP 中所有账户都是合约账户。

2) frozen：该账户是否冻结。

3) votes：查询该地址用于投票的 BHP

4) balance：该地址的资产余额。

5) asset：资产 ID

6) value：资产金额