交易所接入文档

一、 介绍

公信宝(GXB)钱包的对接方式和比特股(BTS)一样，需要运行以下两个程序：见证节点 witness_node 和命令行钱包 cli_wallet。

witness_node 通过 P2P 方式连接到公信链网络，从网络接收最新区块，向网络广播本地签署的交易包。

cli_wallet 通过 websocket 方式连接到 witness_node， 管理钱包文件； 提供交易签名功能，签名后通过 witness_node 向外广播； 通过 http rpc 的方式提供 API 供其他程序调用。

二、 软硬件需求

操作系统： Ubuntu 14.04 LTS 64位系统，内核版本4.4.0-63-generic 以上。

硬件： 内存 8GB+，硬盘50GB+。

依赖：安装 NTP 服务：

```
sudo apt-get install ntp
```

三、 部署和启动程序

1. 安装包下载

选择其中一个下载即可。

平台	下载地址
gith	https://github.com/gxchain/gxb-core/releases/download/1.0.171031/

ub	gxb_1.0.171222.tar.gz
阿里云	http://gxb-package.oss-cn-hangzhou.aliyuncs.com/gxb-core/gxb_1.0.171222.tar.gz

2. 解压程序

将程序放置你的deploy目录，然后执行命令：

```
tar zxvf gxb_1.0.171222.tar.gz
```

3. 启动见证节点witness_node， 同步数据

进入gxb目录，启动公信宝见证节点witness_node

```
# 可以使用2个参数，节省内存： --track-account 和 --partial-operations=true
nohup ./programs/witness_node/witness_node --data-dir=trusted_node --rpc-endpoint="127.0.0.1:28090" \
--p2p-endpoint="0.0.0.0:6789" --log-file --track-account "\"1.2.2999\"" --track-account "\"1.2.3000\"" \
--partial-operations=true --data-transaction-lifetime=1 >>witness.out 2>&1 &
```

witness_node启动参数介绍如下：

```
# 指定数据及配置文件存储的目录
--data-dir=trusted_node

# 指定rpc服务侦听地址及端口(端口可修改)，127.0.0.1限定本地访问rpc服务，若不限定本地访问，可指定0.0.0.0
--rpc-endpoint=127.0.0.1:28090

# 用于连接p2p网络，此参数不建议修改
--p2p-endpoint=0.0.0.0:6789

# 输出日志文件，若无此参数，日志输出到控制台
--log-file
```

```
# 内存中只跟踪指定帐户的交易历史，该选项可传入多次，跟踪多个帐户。请将1.2.2999
替换成交易所的账户数字 ID（在轻钱包账户页面里，账号头像下面会显示一个数字）
--track-account "\"1.2.2999\""

# 每个账户在内存中最多保存NUM条交易记录，默认是全部
--max-ops-per-account=NUM

# 和--track-account / --max-ops-per-account 选项结合，可以进一步节省内
存，建议带上此参数
--partial-operations=true

# 数据交易相关的数据，内存中只保存最近1小时，建议交易所使用此选项，可明显节省内存
占用
--data-transaction-lifetime=1

& 表示程序后台运行

# 重放所有已下载的区块并重建索引，比较耗时
--replay-blockchain

# 删除所有已下载数据，重新同步区块
--resync-blockchain
```

目前全节点程序占用内存20GB+，运行时务必带上--track-account account_id(此处为1.2.x格式的帐户id)和--partial-operations=true参数，内存中只保存交易所帐户的交易历史，内存可以控制在4GB。

```
# 请将1.2.2999 替换成交易所的账户数字 ID ,"1.2."表示类型是账户
--track-account="\"1.2.2999\""
--partial-operations=true
```

完全同步区块，大约需要30分钟以上。通过后台日志文件trusted_node/logs/witness.log可查看区块同步进度，访问公信宝区块浏览器查看最新区块。

区块同步完成后，可以运行命令行钱包cli_wallet。

4. 运行命令行钱包cli_wallet

命令行钱包cli_wallet连接witness_node:

```
./programs/cli_wallet/cli_wallet -s ws://127.0.0.1:28090 \
--enable-rpc-log -r 127.0.0.1:8091 --data-dir=trusted_node
```

cli_wallet启动参数：

```
# 连接见证人节点的websocket rpc地址
-s

# 输出rpc日志文件
--enable-rpc-log

# cli_wallet提供的的websocket rpc地址，开启cli_wallet的 API 服务
# 注意：不要配置为0.0.0.0，因为所有主机都可以访问你的钱包。
-r 127.0.0.1:8091

# 运行在守护进程模式
-d

& 表示程序后台运行
```

首先需要为钱包设置一个钱包密码(这个密码是本地的，用来解锁钱包)：

```
new >>> set_password my_password
# 执行成功后会显示:
locked >>>
# 然后解锁钱包:
locked >>> unlock my_password
# 解锁成功会显示:
unlocked >>>
```

使用 info 命令可以查看当前区块同步情况

```
unlocked >>> info
info
{
  "head_block_num": 3913758,
  "head_block_id": "003bb81eec2abfdb2cf58ffdf4dd547ea190530e",
  "head_block_age": "3 seconds old",
  "next_maintenance_time": "0 second ago",
  "chain_id":
"4f7d07969c446f8342033acb3ab2ae5044cbe0fde93db02de75bd17fa8fd84b8",
```

```
  "participation": "100.00000000000000000",
  ...
}
# head_block_age表示最新的区块时间，系统每3秒出一块
# participation表示见证人参与率，见证人参与率必须大于70，网络才是正常的
```

更多cli_wallet接口，可以查看wallet api说明文档。

5. 设置命令行钱包cli_wallet, 导入帐户私钥

如果你还没有帐户，需要先下载公信宝轻钱包，或者访问在线钱包，注册帐号(记得备份，保存好私钥)。

公信宝轻钱包下载地址	https://www.gxb.io/#downLoad
在线钱包地址	https://wallet.gxb.io

关于私钥和如何使用公信宝钱包，参考 公信宝轻钱包使用手册

在命令行钱包执行命令：

```
// 导入帐户私钥
unlocked >>> import_key account_name wif_key true
```

```
// 查看本钱包能控制的所有帐户
unlocked >>> list_my_accounts
```

```
// 查看帐户信息
unlocked >>> get_account account_name
```

```
// 查看帐户余额
unlocked >>> list_account_balances account_name
```

```
// 转帐(需要帐户有余额)
// 其中GXS代表公信股资产，  GXC代表公信币资产
unlocked >>> transfer from_account to_account 100 GXS "" true
```

```
// 查询最新区块高度， 其中返回结果中head_block_number即最新区块高度
unlocked >>> get_object 2.1.0
```

```
// 查询区块信息，查询时指定区块号
unlocked >>> get_block 881577
```

转帐有2个命令行接口:transfer和transfer2， 其中transfer2执行成功后，返回当前transaction的id

6. 后台运行cli_wallet

在导入钱包私钥完成后，ctrl + c退出，此时会生成本地的钱包文件wallet.json，文件中保存了导入帐户信息、加密的私钥和server地址等信息，cli_wallet重新启动时会加载wallet.json文件。

重新启动cli_wallet，启动时带上参数 -d &， 以daemon模式运行，如下：

```
nohup ./programs/cli_wallet/cli_wallet -s ws://127.0.0.1:28090 \
--enable-rpc-log -r 127.0.0.1:8091 -d >>wallet.out 2>1 &
```

上述命令启动后，需要再多敲一次回车，然后在shell中输入exit来退出终端。相关的控制台输出被重定向到wallet.out文件。

四、 查询账户的交易历史， 监听用户充值

1. 用户向交易所充值

1. 使用公信宝钱包转帐，每一笔交易都可以带一个备注(又称memo)，备注是加密的，只有转帐双方才能解密查看。
2. 交易所可以为每个注册用户分配一个唯一的备注(比如可以使用帐户名、手机号、user id等)，备注长度和内容无限制。
3. 用户向交易所的公信包钱包帐户转帐，带上自己的备注。
4. 交易所通过扫描自己帐户的交易历史，根据转帐的备注，处理用户充值。

5. 已经处理过的交易历史，记录下对应的txID，避免重复入帐。
6. 对于备注为空以及未成功入帐的交易历史，记录下txID，便是后续人工处理工单。

2. 通过账户查询账户的交易历史，以及获取交易的txID

cli_wallet不仅提供了命令行接口，还提供了json rpc接口。 钱包开启了 http rpc 方式的 API 服务时，效果与在钱包里输入命令相同。可以过wscat或者使用http客户端(如curl)来调用。

其中method 传入命令名，params 数组传入参数清单(无参数时，params传空数组)， id为请求的标识，返回结果中的id和请求id一致。如果执行成功，结果会有 result ，否则会有 error

cli_wallet提供了4个查询交易历史的接口：get_account_history、get_relative_account_history、get_account_history_by_operations和get_irreversible_account_history。推荐使用get_irreversible_account_history接口，它只返回不可逆交易历史，并且包含每一笔交易的txID。4个接口在wallet api说明文档里都有说明。以get_irreversible_account_history为例，步骤如下：

1. 解锁钱包。
2. 根据帐户id查询交易历史。

详细过程如下，以帐户gxb-light为例：

1. 解锁钱包:

request：

```
// unlock解锁钱包，其中my_password为解锁密码
curl --data '{"jsonrpc": "2.0", "method": "unlock", "params":
["my_password"], "id": 1}' http://127.0.0.1:8091/rpc
```

解锁成功，返回：

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

2. 调用钱包的get_irreversible_account_historys接口， 查询帐户

的不可逆交易历史，接口返回信息包含txID:

若没有调用unlock解锁钱包，则查询出的转帐交易memo是无法解密的。此处，只有交易双方才能解密memo。

request:

```
// 调用参数：get_irreversible_account_history <account_name>
<[operation types]> <start> <limit>
// <account_name> 为帐户名或者帐户id，帐户名需要为小写
// <[operation types]> 为筛选用的operation type数组，只查询指定类型的交易
历史，若传入空[]，则表示获取全部类型的交易历史；如果只查询转帐交易历史，可以传
入[0]
// <start> 为起始序号，最早一笔交易历史序号为1
// <limit> 为limit，表示获取limit笔交易历史
// get_irreversible_account_history接口返回结果中，包含详细交易历史，和下
一次翻页查询时的起始序号(start参数)
curl --data '{"jsonrpc": "2.0", "method":
"get_irreversible_account_history", "params": ["gxb-light",[0], 1,
10], "id": 1}' http://127.0.0.1:8091/rpc
```

response:

```
{
  "id": 1,
  "jsonrpc": "2.0",
  "result": {
    "next_start_sequence":28, // 下一次调
用get_irreversible_account_history时，需要传入的起始序号(start参数)
    "result_count":10, // 筛选后实际返回的记录数量，若小于limit，则表示已经
查询到最新的交
    "details": [{ //  返回的details是一个数组，每个item是一笔交易
      "memo": "196702323",  // 此为解密过的memo，只有交易双方才能解密memo
      "description": "Transfer 3099 GXS from gxb-light to yunbi-gxs
-- Memo: 196702323   (Fee: 0.09493 GXS)", //交易描述
      "op": {  // 交易中包含的操作
        "id": "1.11.3436",
        "op": [0, {// 序号0表示转帐
          "fee": {
            "amount": 9493,// 操作的手续费为(9493/100000) GXS，数字要除
以100000
            "asset_id": "1.3.1" //  资产1.3.1是GXS
          },
```

```
          "from": "1.2.3054",// 转帐的发起帐户
          "to": "1.2.2999",// 转帐的接收帐户
          "amount": {
            "amount": 300000,  // 转帐金额为(300000/100000) 即3个GXS
, 数字要除以100000
            "asset_id": "1.3.1"
          },
          "memo": {
            "from":
"GXC68o9LkFKv5ihSt6z9oTmc7wVALUmT5Kd75BTy9rMp38wSuWU5N",
            "to":
"GXC5yRwAseZhPBorMmxXhQvmvsDzoJJbQnau5fFboGr5V2zGZPWsk",
            "nonce": "384187316390066",
            "message": "1c05a47a362361c0cf2dba52d08f3517"  // 此为加
密的memo
          },
          "extensions": []
        }],
        "result": [0, {}],
        "block_num": 1115034, // 交易被打包进哪个区块，所属的区块号(区块高
度)
        "trx_in_block": 0,// 交易在区块中的索引位置
        "op_in_trx": 0,
        "virtual_op": 3482
      },
      "transaction_id": "f9f8f8359c59ac1341516facdf30c98fd5d57b5b"
    }, {
      "memo": "",
      "description": "Transfer 7100 GXS from yunbi-gxs to gxb-light
(Fee: 1 GXC)",
      "op": {
        "id": "1.11.3331",
        "op": [0, {
          "fee": {
            "amount": 100000,
            "asset_id": "1.3.0"
          },
          "from": "1.2.2999",
          "to": "1.2.3054",
          "amount": {
            "amount": 710000000,
            "asset_id": "1.3.1"
          },
          "extensions": []
        }],
        "result": [0, {}],
        "block_num": 1000766,
        "trx_in_block": 0,
```

```
        "op_in_trx": 0,
        "virtual_op": 3377
      },
      "transaction_id": "7c64c51ee931043ca1bc6791efc942e94b8236af"
    }]
  }
}
```

注意：对交易所来说，转帐操作的资产id必须要是1.3.1(GXS)。

不可回退区块

调用cli_wallet的get_dynamic_global_properties接口，查看当前最大的不可回退区块号(也即最大的不可回退区块高度)。小于此区块高度的区块，其包含的交易都是已经被确认不可回退的。

cli_wallet提供的get_irreversible_account_history接口，返回的结果是不可回退的，即最终被确认过的交易。

```
curl --data '{"jsonrpc": "2.0", "method":
"get_dynamic_global_properties", "params": [], "id": 1}'
http://127.0.0.1:8091/rpc
```

返回结果：

```
{
  "id": 1,
  "result": {
    "id": "2.1.0",
    "head_block_number": 1724515, // 当前最大区块号(区块高度)
    "head_block_id": "001a506300f6717c3e99b6d8d89b264d94f1793c",
    "time": "2017-08-16T14:51:53",
    "current_witness": "1.6.2",
    "next_maintenance_time": "2017-08-16T15:00:00",
    "last_budget_time": "2017-08-16T14:50:00",
    "witness_budget": 0,
    "accounts_registered_this_interval": 0,
    "recently_missed_count": 1,
    "current_aslot": 2426557,
    "recent_slots_filled":
"326968992855967788277935148493325729655",
    "dynamic_flags": 0,
```

```
    "last_irreversible_block_num": 1724507   // 最大的不可回退区块号(区块高度)为1724507
  }
}
```

五、常见问题

1. **区块同步时报错**

   观察后台日志文件trusted_node/logs/witness.log, 如果日志持续报错，如"unlinkable block", "block does not link to known chain"，这是区块同步出错了。

   解决方法：

   i. 区块同步异常，有可能本地的区块链文件坏掉了。需要停止witness_node程序，然后删除trusted_node， 重新启动witness_node。

   ii. 或者不需要删除trusted_node目录，启动命令加上参数--resync-blockchain， 会重新同步区块。

2. **如何正常关闭witness_node**

   i. 如果witness_node没有后台运行，则执行一次Ctrl + C, 然后等待程序保存内存数据后自动退出。

   ii. 如果witness_node运行在后台， 执行 `kill -s SIGINT $(pgrep witness_node)`，等待程序保存内存数据后自动退出。不能使用kill -9， 否则下次启动会重建索引，启动比较慢。

3. witness_node重启以后，需要重新启动cli_wallet。因为cli_wallet后台运行时，不会自动退出。关闭cli_wallet的方法：执行 `kill -s SIGINT $(pgrep cli_wallet)`

4. 如果异常退出，则重新启动时，很可能需要重建索引，启动比较慢。如果witness_node 出现异常，一般先尝试带--replay-blockchain 参数重启，即手工触发重建索引。

5. witness_node重启后，需要重启cli_wallet。如果cli_wallet是后台运行的，cli_wallet不会因witness_node退出而自动退出，也需要重启。

6. **cli_wallet进程偶尔退出问题**

   如果遇到cli_wallet后台运行一段时间后退出的情况，可能的原因是终端掉线，建议后台运行成功后，关闭当前终端。或者在启动命令行之前加上nohup:

```
nohup ./programs/cli_wallet/cli_wallet -s ws://127.0.0.1:28090  --
enable-rpc-log -r 127.0.0.1:8091 -d >>wallet.out 2>1 &
```

7. cli_wallet连接witness_node时，返回"Timer Expired"连接失败。 原因是开启了ufw，可以关闭ufw并reboot， 或者自行配置ufw。

8. 建议使用脚本启动程序(脚本中的帐户id需要做修改)： witness_node启动脚本 和 cli_wallet启动脚本

六、注意事项

1. 用户充值。每笔转账可以带一个备注(memo)，交易所通过这个备注来区分是哪个用户的充值。具体备注与交易所用户关联关系，请交易所自行设定。备注是加密的，只有转帐双方才可以解密。
2. 转帐手续费问题。转帐的手续费由2部分组成：基本手续费 + memo费用，其中基本手续费为0.05GXS, memo费用按总字节长度收费，0.5GXS/`KB`。一次转帐手续费计算：`0.05 + 0.5*(KBs of memo) GXS ， 取KB时截断取整`。 带备注的转帐，手续费一般在[0.05, 0.15] GXS之间(假设备注长度不超过100字节)。
3. 交易所查询交易历史，处理用户充值时，需要处理用户充值时填入memo的特殊情况。比如交易所提供给用户的memo是1234567，而用户充值填写的memo是"我的memo是1234567!"，类似情况需要考虑。
4. 不建议交易所提供类似GXS123456的memo，把memo作为帐户进行充值。**memo长度和内容无限制，建议memo长度超过10位，并且以数字开头。**
5. 钱包的json rpc接口，帐户名需要传入小写，不能有大写字母。用户提现到公信宝钱包时，需要将绑定的帐户名转为小写。
6. **人工处理用户充值问题**：未正确入帐的交易信息，交易所需要妥善保存下来，以备客服人工入帐。用户提交工单时，需要用户提供txID(用户可以通过钱包查看当前转帐信息获得)，txID可以保证用户充值转帐的唯一性。务必注意：**平台保存好人工处理过的txID, 同一txID只处理1次**。如果不同的注册用户使用同一txID提交工单的，属于欺骗!
7. 系统中存在多种资产，其中资产id（asset_id）1.3.1为GXS。监听用户充值时，请务必校验转帐交易中的asset_id字段为1.3.1。

8. 用户提现。调用transfer/transfer2处理用户提现时，转帐数量请传入字段串，加双引号。如下：

```
curl --data '{"jsonrpc": "2.0", "method": "transfer2", "params":
["from_account", "to_account", "100.01", "GXS", "",  true], "id":
1}' http://127.0.0.1:8091/rpc
```

9. GXS精度，为小数点后5位，即最小单位为0.00001 GXS。公信链中没有小数，数字在系统中被放大了10万倍，所以get_irreversible_account_history接口返回的数字，比如转帐的数量，需要除以10万，才是真正的数量。
10. 钱包状态为locked状态时只能查询，不能转帐，不能解密转帐备注。如果需要转帐或者查询交易历史，需要先unlock。
11. **transfer2转帐时，第3个参数转帐数量如果包含小数，必须加双引号，否则转帐会失败。建议转帐数量统一加上双引号。**

相关文档：

1. witness_node启动脚本
2. cli_wallet启动脚本
3. 备用cli_wallet启动脚本，脚本提供3个主网接入点，如果本地witness_node暂时不可用，可以执行此脚本，连接主网接入点
4. 公信宝冷钱包离线签名教程
5. wallet api说明文档