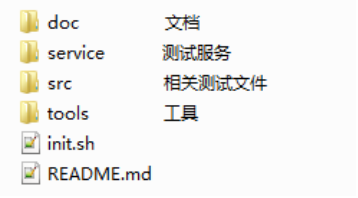
NEO python 测试框架 api 及测试用例详细介绍

一、 客户端测试框架及测试用例详细介绍：

在整体的 neo 文件夹中包含的文件如下：



其中的 tools 文件夹中包含一个 init_selfchecker.py 的文件，此文件为自检文件，该文件中包含的方法为 stop_nodes()、start_nodes()、clear_nodes()、copy_nodes()、check_connected_nodes()、check_all()等方法，用于停止和启动节点、清除和复制节点、检测节点间的连接和整体自检。此文件具体如何使用及测试框架请参考另外一份文档《Neo python 测试框架基础使用介绍》。

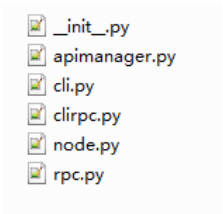
其中的 init.sh 是对整体测试框架 api 的初始化文件。

其中的 doc 文件夹包含 NEO_cli 的测试框架和测试用例，TN_neo_cli 文件以及 NEO 测试框架 api 及测试用例的文件说明。

在 src 下包含文件如下：

1. 其中 api 和 utils 中包含了测试脚本相关 API 和测试框架相关代码：

   1) api 中目录如下：



apimanager.py、cli.py、clirpc.py、node.py、rpc.py 分别为 apimanager，node，rpc 和 cli 相关 API，apimanager.py 整合了这些 API，通过 from api.apimanager import API 即可调用以上四个文件中的方法。

a) node.py：其中包含 setnode()、simplerun()、sftp_transfer()、clear()、send_file()、wait_gen_block()， wait_tx_result()，get_current_note()， check_node_block()，relace_configs()，relace_config()，exec_cmd()等方法，用于设置节点、运行节点服务器、转移节点、清除节点、发送节点文件、实现等待区块生成、等待区块生成结果、获取当前节点地址、检查节点服务器数据库是否一致、替换文件相关 configs 和 config、调用方法返回 " True " 值。

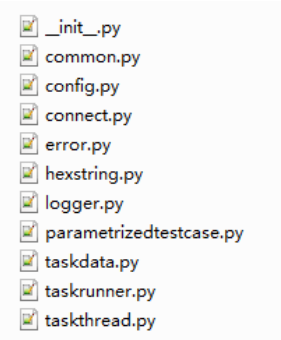
b) rpc.py：其中包含 setnode()，simplerun()，dumpprivkey()，getaccountstate()，getassetstate()，getbalance()，getbestblockhash()，getblock()，getblockcount()，getblockheader()，getblockhash()，getblocksysfee()，getconnectioncount()，

getcontractstate()，getnewaddress()，getrawmempool()，getrawtransaction()，getstorage()，gettxout()，getpeers()，getvalidators()，getversion()，getwalletheight()，invoke()，invokefunction()，invokescript()，listaddress()，sendfrom()，sendrawtransaction()，sendtoaddress()，sendmany()，validateaddress()等方法，用于设置节点、运行节点、及封装了被测对象的 rpc 接口，方便调用。

c) cli.py：其中包含两个类分别是 CLIReadThread 和 CLIApi。CLIReadThread 类中包含 lines()，isfinish()，isblock()，clear()，run()，CLIApi 类中包含 init()，readmsg()，clearmsg()，begincmd()，endcmd()，writeline()，writesend()，writeexcept()，terminate()，waitnext()、waitsync()、exec()、version()，help()，wait()，clear()，exit()，create_wallet()，open_wallet()，upgrade_wallet()，rebuild_index()，list_address()，list_asset()，list_key()，show_utxo()，show_gas()，claim_gas()，create_address()，import_key()，export_key()，send()，import_multisigaddress()，sign()，relay()，show_state()，show_node()，show_pool()，export_all_blocks()，export_blocks()，start_consensus()等方法，CLIReadThread 的类中封装了 cli 读取线程的方法，CLIApi 的类中封装了 cli 的执行命令及相关处理方法。此方法是在本地内部服务器之间进行的调用测试。（关于 CLIApi 种方法的具体内容，参照“二测试服务端及相关工具文件”中的第 2 点）

d) clirpc.py：其中包含 setnode()、simplerun()、init()、readmsg()、clearmsg()、terminate()、version()、help()、wait()、clear()、exit()、create_wallet()、open_wallet()、update_wallet()、rebuild_index()、list_address()、list_asset()、list_key()、show_utxo()、show_gas()、claim_gas()、create_address()、import_key()、export_key()、send()、import_multisigaddress()、sign()、relay()、show_state()、show_node()、show_pool()、export_all_blocks()、export_blocks()、start_consensus()等方法，这些方法封装了 cli 的执行命令及相关处理方法。此方法可以通过启动远程共识，可以检测到对外部远程客户端的调用情况。（关于方法的具体内容，参照“二测试服务端及相关工具文件”中的第 2 点）

2) utils 中目录如下：



a) common.py 中包含 cmp()，pause()，bl_reserver()，base58_to_address()，address_to_base58()，bl_address()等方法，这些方法用于实现比较 str、dict、list 数据是否一致，中止测试，base58 编码和 bl_reserver 的字符串两两转序。

b) config.py 中包含了 case 中及测试代码中会使用到的部分变量和路径。

c) connect.py 中包含类 WebSocket()，在这个类中存在一个名叫 exec 的方法将其内线程 ws_thread 和 ws_heartbeat_thread 打开，然后监听输入内容并根据内容发送请求。除此以外还存在以下三个方法：

i. con(itype,ip,request) ：

根据 itype（可以为"RPC","ST"）选择调用接口。itype 为"RPC"，"RESTFUL"，"WS"和"ST"时分别调用 con_rpc 和 con_test_service。

ii. con_rpc(ip,request)：若参数 ip 存在输入，url 为"http://"+ip+"/jsonrpc"，若没有则从上层目录中的 config.json 中读取 url，从该 url 读取 response 返回，ip 端口号为 10003、20003、30003、40003、50003。itype 为"RPC"时调用。

iii. con_test_service(ip,request)：同上用途，但 ip 端口号为 23606、23616、23626、23636、23646。itype 为"ST"调用。

d) error.py 用于捕捉测试用例时可能遇到的异常。

e) hexstring.py 文件中主要包含了一些数据转换的方法：

i. FileToHex(path)：将参数 path 路径中文件内容 ASCII 值对应的十六进制数作为返回值。

ii. ByteToHex(bytes_value)：将参数 bytes_value 中内容 ASCII 值对应的十六进制数作为返回值。

iii. HexToByte(hexStr)：将参数 hexStr 中的十六进制数转换为对应 ASCII 值。

iv. recoveryAddress(m,hexArr,checkMultiSig="ae")：返回原 pubkey 经过一系列处理后的散列值（最后返回结果为 40 位十六进制数），同时 print sha256 与 ripemd160 的值。

v. numSeq(m)：当 m 为数字 0 时 return "00"，当 m 为大于等于 1 且小于等于 16 的正整数时返回八十加 m 的十六进制数，当 m 大于 16 后返回（m+1）的十六进制后两两倒序排列后的数据。

f) logger.py 用于记录 log 信息。

g) parametrizedtestcase.py 用于实现参数化。

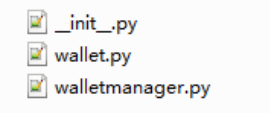
h) taskdata.py 类 Task 用于返回 task 中相应的数据；类 TaskData 用于将 path 路径下.json 文件中的数据作为 task 添加到 ret，最后返回 ret。

i) taskrunner.py 包含两个基础的方法 run_single_task()和 run_pair_task()，前者用于执行单个 webapi 并将返回的 response 与期望 response 作对比，一致时输出 OK 相关语句，不一致时输出 Failed 相关语句（包括执行所使用时间），最终返回 result 和 response；后者当参数 compare_src_key 为 bool 型的 True 时返回调用 run_single_task(task1)和 run_single_task(task2)结果 response 的比较结果。

j) taskthread.py 用于初始化进程，也包括启动进程和获取结果。

2. monitor 中包含了一个 monitor.py 文件，该文件用于检测和恢复测试环境（包括重启各节点服务器和签名服务器 sigserver），在每次执行测试脚本时会调用 monitor 以保证测试环境。当使用调用 run_alltest.py 时-m 后跟 0 时不再调用 monitor。

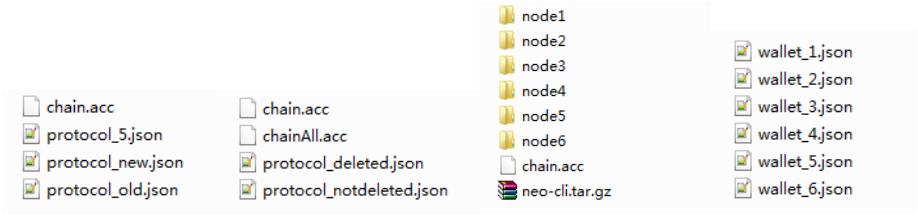
3. neo 中包含钱包控制文件，目录如下所示：



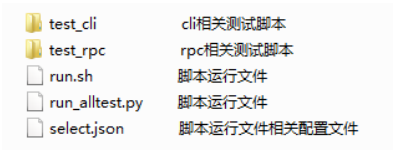
其中 wallet.py 和 walletmanager.py 两个文件，walletmanager 整合了 wallet.py 中的方法，通过 from neo.wallet import wallet 即可调用其中的方法，以下是 wallet.py 中的方法.

Wallet.py：包含了两个类，分别是 Wallet 类和 Account 类，Wallet 类中包含的方法为 path()、name()、version()、scrypt_n()、scrypt_r()、scrypt__p()、account()，Account 类中包含的方法为 address()、label()、isDefault()、lock()、key()、contact()。通过调用方法从而检测到调用的具体某个.json 文件服务器，从而达到控制钱包的作用。

4. resource 包含了自检文件中用于参考的各类文件，其中包含 consensus，exportblocks，wallet 和 nodes 四个文件夹，consensus 文件夹中包含测试用例所需的启动共识节点的相关文件，exportblocks 文件夹中包含进行离线同步操作的具体相关文件，wallet 文件夹中包含六个需要被使用的.json 的钱包文件,nodes 文件夹中包含六个对应六个.json 文件的使用节点文件，其目录如下所示：



5. test 中放置了所有测试脚本：



1) 所有测试文件中框架如下，其中 resource 和 test_api 根据测试实际情况会存在出入：

```
test_xxxx
    |--resource   ##一般包含合约.neo 文件和 neo 相关配置文件
    |--__init__.py
    |--test_config.py   ##包含 test_main 所用到的配置
    |--test_main.py   ##包含所有相关测试用例
```

2) 以下是测试文件与包含用例的对应关系，测试文件 test_main.py 中测试用例的命名如下例子：test_01_createwallet 其中 test 用来表示测试用例，数字 01 与表格中测试用例序号相对，createwallet 一般为测试的方法名（部分为实际测试文件中调用的方法名）。

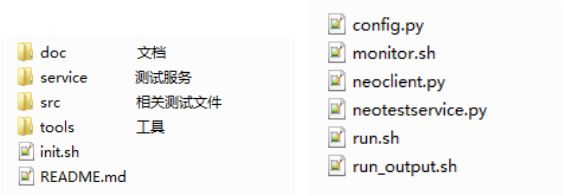
   a) test_cli 中的 case 对应 neo 自动测试工具中的打开钱包并开启转账功能管理的相关执行命令及相关工具。

   b) test_rpc 中的 case 对应 neo 自动测试工具中的 RPC API。

   c) run_alltest.py 以及 select.json 通过读 config 文件和《Neo python 测试框架基础使用介绍》，来获取测试用例和详细功能介绍，run.sh 脚本运行文件，是用于启动测试服务的文件。

二、 测试服务端及相关工具文件

测试服务端的文件在 neo 文件夹下的 service 文件夹中，目录如下图所示：



1. 测试脚本及相关文件已在第一节客户端测试框架及测试用例中详细介绍，此处不再赘述。
2. 其中的 monitor.sh 文件，该文件用于启动错误检测与恢复功能，在运行测试用例的时候自动开启，错误堆积一定程度集中清理，确保程序不会卡顿。
3. 其中的 neoclient.py 是一个 cli 的 example 文件，用于在程序中进行测试的某一案例使用，用于编写程序过程中的检验的 example 文件。
4. 测试服务工具中的服务会在测试前配置好环境以及启动测试服务（见第四节第三点）。其中的 install.sh 用于安装所需要的 python 包，配置文件 config.py 中包含了 PORT，run.sh 和 run_output.sh 用于启动测试服务，neotestservice.py 即为 run.sh 所启动的测试服务。启动该服务后方可实现多签和请求其他节点数据库数据。Neotestservice.py 文件是整合 api 文件夹中的 cli.py 的方法并进行整体调用。neotestservice.py 包含方法如下：
    1) get_host_ip()：调用后返回主机地址。
    2) eval_exceptfunc(funcstr)：执行参数中包含的命令，如果"funcstr"为空值则返回空值，否则返回"funcstr"的值。
    3) exec_cmd(**kwargs)：执行参数中所包含的命令，结束后返回 True。
    4) cli_init(**kwargs)：调用后初始化脚本。
    5) cli_readmsg(**kwargs)：执行参数中所包含的命令，结束后返回读写器中内容。
    6) cli_clearmsg(**kwargs)：调用后返回清除消息信息。
    7) cli_terminate(**kwargs)：返回终止指令。
    8) cli_exec(**kwargs)：返回执行指令。
    9) cli_version(**kwargs)：返回版本信息。
    10) cli_help(**kwargs)：调用帮助菜单指令。
    11) cli_clear(**kwargs)：调用清除屏幕指令。
    12) cli_exit(**kwargs)：调用退出程序指令。
    13) cli_create_wallet(**kwargs)：创建钱包文件。
    14) cli_open_wallet(**kwargs)：打开钱包文件。
    15) cli_upgrade_wallet(**kwargs)：升级旧版钱包文件。
    16) cli_rebuild_index(**kwargs)：重建钱包索引，这一步骤需要打开钱包文件。
    17) cli_list_address(**kwargs)：列出钱包中的所有帐户，这一步骤需要打开钱包文件。
    18) cli_list_asset(**kwargs)：列出钱包中的所有资产，这一步骤需要打开钱包文件。
    19) cli_list_key(**kwargs)：列出钱包中的所有公钥，这一步骤需要打开钱包文件。
    20) cli_show_utxo(**kwargs)：列出钱包中指定资产的 UTXO，这一步骤需要打开钱包文件。
    21) cli_show_gas(**kwargs)：列出钱包中的所有可提取及不可提取的 GAS，这一步骤需要打开钱包文件。
    22) cli_claim_gas(**kwargs)：提取钱包中的所有可提取的 GAS，这一步骤需要打开钱包文件。
    23) cli_create_address(**kwargs)：创建地址/批量创建地址，这一步骤需要打开钱包文件。
    24) cli_import_key(**kwargs)：导入私钥/批量导入私钥，这一步骤需要打开钱包文件。

25) cli_export_key(**kwargs)：导出私钥，这一步骤需要打开钱包文件。
26) cli_send(**kwargs)：向指定地址转账，参数分别为：资产 ID、对方地址、转账金额、手续费，这一步骤需要打开钱包文件。
27) cli_import_multisigaddress(**kwargs)：创建多方签名合约，这一步骤需要打开钱包文件。
28) cli_sign(**kwargs)：签名 参数为：记录交易内容的 json 字符串，这一步骤需要打开钱包文件。
29) cli_relay(**kwargs)：广播 参数为：记录交易内容的 json 字符串，这一步骤需要打开钱包文件。
30) cli_show_state(**kwargs)：显示当前区块链同步状态。
31) cli_show_node(**kwargs)：显示当前已连接的节点地址和端口。
32) cli_show_pool(**kwargs)：显示内存池中的交易（这些交易处于零确认的状态）。
33) cli_export_all_blocks(**kwargs)：导出全部区块数据，导出的结果可以用作离线同步。
34) cli_export_blocks(**kwargs)：从指定区块高度导出指定数量的区块数据，导出的结果可以用作离线同步。
35) cli_start_consensus(**kwargs)：启动共识。
36) application(request)：调用以上的 dispatcher，返回调用结果，并且在没有 error 时在结果中添加"error":0，在有 error 时在结果中添加"desc":error_message 和"error":error_code