Neuralium Command Line Interface (neuralium-cli)

Release 1.0.0

Neuralium Inc.

Jan 03, 2021

CONTENTS:

CHAPTER

ONE

USAGE

The Neuralium Cli's documentation can be obtained from the `help` command

```
$ ./neuraliumcli help
Usage:
    <Command$

Required Parameters:
    <Command$
        CanPublishAccount - Can we publish our account? (bool)
        CompleteLongRunningEvent - Wait for a long running event's completion (bool)
        CreateNewWallet - Creates a new wallet [long-running] (int)
        EnterKeyPassphrase - Key's passphrase
        EnterWalletPassphrase - Wallet's passphrase
        exit - Exits the shell
        help - Help [Alias: h]
        IsMiningEnabled - Are we currently mining? (bool)
        IsWalletLoaded - Is the wallet loaded? (bool)
        LoadWallet - Loads the wallet [long running] (int or object)
        Ping - Pings the server (bool)
        PresentAccountPublicly - Send the account presentation transaction [long-
↪running] (int)
        PublishAccount - Send the account publication transaction [long-running] (int)
        QueryAccountTotalNeuraliums - Query an account's total amount of Neuraliums␣
↪(object)
        QueryBlock - Query a block's details (string)
        QueryBlockBinaryTransactions - Query block's transaction in binary format␣
↪(List<object$)
        QueryBlockChainInfo - Query the blockchain's info (object)
        QueryBlockHeight - Query what is the current block height (long)
        QueryChainStatus - Query the chain's status (object)
        QueryCompressedBlock - Query a block's compressed details (byte[])
        QueryElectionContext - Query a block's election context (object)
        QuerySupportedChains - Query what chain we support (object)
        QueryWalletAccounts - Query the wallet's accounts (List<object$)
        QueryWalletTransactionHistory - Query the wallet's transactions history (List
↪<object$)
        RenewLongRunningEvent - Renew a long running event (bool)
        run - Operation [Alias: r]
        SendNeuraliums - Send neuralium to another account [long-running] (int)
        Shutdown - Send a shutdown request to the server (bool)
        StartMining - Starts the mining
        StopMining - Stops the mining
        Test - Tests the server
        WalletExists - Can the wallet be found? (bool)
```

(continues on next page)

(continued from previous page)

```
        WalletKeyFileCopied - Wallet's Key file is copied
```

1.1 Usage instructions.

There are two modes of operation, the **interactive mode** and the **stateless mode**. To enter in the interractive mode, simply do

```
$ ./neuraliumcli
> [you are now in interactive mode, you can enter any command, including 'help']
```

otherwise you can use the "stateless mode"

```
$ ./neuraliumcli help
[...]
```

1.2 Examples

The following examples use the `stateless mode` but the syntax is the same for the `interractive mode`, omitting the `./neuraliumcli` of course.

1.2.1 Json parameters

$ ./neuraliumcli run SendNeuraliums jparams='[{"Name":"targetAccountId", "Value":"{SF3}"},{"Name":"amount", "Value":"1.1"}]'

1.2.2 Ordered Parameters List

*$ ./neuraliumcli SendNeuraliums {SF3} 1.1 0.001

1.2.3 Example commands

Run individual commands:

$ ./neuraliumcli Ping

$ ./neuraliumcli QueryBlockChainInfo

$ ./neuraliumcli IsWalletLoaded

$ ./neuraliumcli WalletExists

$ ./neuraliumcli LoadWallet

$ ./neuraliumcli CreateNewWallet "test" 1 false false false {} false

$ ./neuraliumcli run CreateNewWallet jparams='[{"AccountName":"account name","AccountType":1,"EncryptWallet":false,"Encrypt
"Passphrases":{}, "PublishAccount":false}]'

$ ./neuraliumcli QueryBlock 108

$ ./neuraliumcli QueryWalletAccounts

$ ./neuraliumcli QueryDefaultWalletAccountId

$ ./neuraliumcli QueryDefaultWalletAccountUuid

$ ./neuraliumcli run QueryAccountTotalNeuraliums jparams='[{"AccountUuid"=016d2570-7b0a-44dd-b410-bb479776d6c2}]'

$ ./neuraliumcli run PublishAccount jparams='[{"AccountUuid":016d4f5a-c134-4141-8e35-3278b9baed67]'

$ ./neuraliumcli run SendNeuraliums jparams='[{"TargetAccountId":"{SF3}","Amount":1.1,"Tip":0.001,"note":"some note"}]'

$ ./neuraliumcli StartMining

$ ./neuraliumcli StopMining

$ ./neuraliumcli Shutdown

CHAPTER

TWO

API PAGE

2.1 CanPublishAccount

```
neuraliumcli.dll CanPublishAccount <account-code> [--config=<str>]
    [--host=<str>] [--port=<int>] [--runtime-mode=<str>]
```

Can we publish our account? (bool)

2.1.1 Return value

`true` is the account is ready for publication, `false` otherwise.

2.2 CompleteLongRunningEvent

```
neuraliumcli.dll CompleteLongRunningEvent <correlation-id> [<data>] [--config=<str>]
    [--host=<str>] [--port=<int>] [--runtime-mode=<str>]
```

Triggers the completion of a long running event (bool)

2.2.1 Parameters

1. CorrelationId (Required, Unsigned Interger): The correlation id returned by the long running task that you started (e.g. *CreateNewWallet*)

2.2.2 Return Value

`true` if operation was successful

2.3 CreateNewWallet

```
neuraliumcli.dll CreateNewWallet <account-name> <account-type> <encrypt-wallet>
    <encrypt-key> <encrypt-keys-individually> <passphrases> <publish-account>
    [--config=<str>] [--host=<str>] [--port=<int>] [--runtime-mode=<str>]
    [--timeout=<double>]
```

Allows you to create a new wallet, and potentially to publish it [long-running] (uint)

2.3.1 Parameters

1. AccountName (Required, String): a name for your account (e.g. "MyNeuraliumAccount")
2. AccountType (Required, integer): the type of account
   - User = 1,
   - Server = 2,
   - Moderator = 3 (reserved),
   - Joint = 4,
3. EncryptWallet (Required, boolean): wheter to encrypt the wallet or not
4. EncryptKey (Required, boolean): wheter to encrypt the keys separately or not
5. EncryptKeysIndividually (Required, boolean): wheter to encrypt each kind of key with different passwords
6. Passphrases (Required, list of *key="value"* pairs): the passphrases dictionnary, depending on your encryption options, you will use a different subset of indices
   - index **0**: the *wallet* password
   - index **1**: the *transactions key* password, or the password for all keys if *EncryptKeysIndividually=false*
   - index **2**: the *messages key* password (only used if *EncryptKeysIndividually=true*)
   - index **3**: the *key change key* password (only used if *EncryptKeysIndividually=true*)
   - index **4**: the *super key* password (only used if *EncryptKeysIndividually=true*)
   - index **5**: the *validator signature key* password (only used if *EncryptKeysIndividually=true*)
   - index **6**: the *validator secret key* password (only used if *EncryptKeysIndividually=true*)
   - [**Usage**] If you enable only *EncryptWallet*, the usage would be
     - `./neuraliumcli CreateNewWallet "My_User_Account" 1 true false false 0="my wallet password" false`
   - [**Usage**] If you enable *EncryptWallet* and *EncryptKey*, the usage would be
     - `./neuraliumcli CreateNewWallet "My_Server_Account" 2 true true false '0="my wallet password",1="my keys password"' false`
7. PublishAccount (Required, boolean): whether to proceed with account publication immediately

2.3.2 Return Value

The operation returns a correlation id (an integer) that you can use to track progress of wait for completion, using other commands such as *CompleteLongRunningEvent*

2.4 EnterKeyPassphrase

```
neuraliumcli.dll EnterKeyPassphrase <correlation-id> <key-correlation-code>
    <passphrase> [--config=<str>] [--host=<str>] [--port=<int>] [--runtime-mode=<str>]
```

Key's passphrase

2.4.1 Parameters

1. CorrelationId (Required, Unsigned Interger): The correlation id returned by the long running task that you started (e.g. *CreateNewWallet*)
2. KeyCorrelationId (Required, Unsigned Interger): The key correlation id returned by the long running task that you started (e.g. *CreateNewWallet*)
3. Passphrase (Optional, String): the passphrase, if omitted, it will be asked from standard input

2.5 EnterWalletPassphrase

Same as *EnterKeyPassphrase* but for wallet.

2.6 IsMiningEnabled

```
    neuraliumcli.dll IsMiningEnabled [--config=<str>] [--host=<str>] [--port=<int>]
        [--runtime-mode=<str>]
```

Are we currently mining? (bool)

2.6.1 Return Value

`true` if mining is enabled.

2.7 IsWalletLoaded

```
neuraliumcli.dll IsWalletLoaded [--config=<str>] [--host=<str>] [--port=<int>]
    [--runtime-mode=<str>]
```

Is the wallet loaded? (bool)

2.7.1 Return Value

`true` if a wallet is loaded.

2.8 LoadWallet

```
neuraliumcli.dll LoadWallet [<passphrase>] [--config=<str>] [--host=<str>]
    [--port=<int>] [--runtime-mode=<str>] [--timeout=<double>]
```

Loads the wallet [long running] (uint or object)

2.8.1 Parameters

1. Passphrase (Optional, String): The passphrase. If omitted, it will be asked from standard input.

2.8.2 Return Value

The correlation id (Unsigned Interger)

2.9 Ping

```
neuraliumcli.dll Ping [--config=<str>] [--host=<str>] [--port=<int>]
    [--runtime-mode=<str>]
```

Pings the server (bool)

2.9.1 Return Value

`true` if your instance of the neuralium node answered.

2.10 PresentAccountPublicly

```
neuraliumcli.dll PresentAccountPublicly [--config=<str>] [--host=<str>] [--port=<int>]
    [--runtime-mode=<str>]
```

Sends the account presentation transaction [long-running] (uint)

2.10.1 Return Value

The correlation id (Unsigned Interger)

2.11 PresentAccountPublicly

```
neuraliumcli.dll PublishAccount <account-code> [--config=<str>] [--host=<str>]
    [--port=<int>] [--runtime-mode=<str>]
```

Sends the account publication transaction [long-running] (uint)

2.11.1 Return Value

The correlation id (Unsigned Interger)

2.12 QueryAccountTotalNeuraliums

```
neuraliumcli.dll QueryAccountTotalNeuraliums <account-code> [--config=<str>]
    [--host=<str>] [--port=<int>] [--runtime-mode=<str>]
```

Query an account's total amount of Neuraliums (object)

2.12.1 Parameters

1. AccountCode (Required, String): The account code

2.12.2 Return Value

The amount of neuraliums.

2.13 QueryBlock

```
neuraliumcli.dll QueryBlock <block-id> [--config=<str>] [--host=<str>] [--port=<int>]
    [--runtime-mode=<str>]
```

Query a block's details (string)

2.13.1 Parameters

1. BlockId (Required, Interger): the desired block's id

2.13.2 Return Value

The block's details (json)

2.14 QuerySupportedChains

```
neuraliumcli.dll QuerySupportedChains [--config=<str>] [--host=<str>] [--port=<int>]
    [--runtime-mode=<str>]
```

Query what chain we support (object)

2.14.1 Return Value

The chain support level deatils (json)

2.15 QueryBlockHeight

```
neuraliumcli.dll QueryBlockHeight [--config=<str>] [--host=<str>] [--port=<int>]
    [--runtime-mode=<str>]
```

Query what is the current block height (long)

2.15.1 Return Value

An Integer, the current block height of the blockchain.

2.16 QueryChainStatus

```
neuraliumcli.dll QueryChainStatus [--config=<str>] [--host=<str>] [--port=<int>]
    [--runtime-mode=<str>]
```

Query the chain's status (object)

2.16.1 Return Value

A `json` dictionnary with various informations. For example:

```
{
  "walletInfo": {
    "walletExists": true,
    "walletFullyCreated": true,
    "isWalletLoaded": true,
    "walletEncrypted": false,
    "walletPath": "/path/to/your/.neuralium/neuralium"
  },
  "minRequiredPeerCount": 1,
  "miningTier": 255
}
```

2.17 QueryCompressedBlock

```
neuraliumcli.dll QueryCompressedBlock <block-id> [--config=<str>] [--host=<str>]
    [--port=<int>] [--runtime-mode=<str>]
```

Query a block's compressed details (byte[])

2.17.1 Parameters

1. BlockId (Required, Interger): the desired block's id

2.17.2 Return Value

The block's compressed bytes (byte[])

2.18 QueryBlockChainInfo

```
neuraliumcli.dll QueryBlockChainInfo [--config=<str>] [--host=<str>] [--port=<int>]
    [--runtime-mode=<str>]
```

Query the blockchain's info (object)

2.18.1 Return Value

The blockchain information (json)

2.19 QueryBlockBinaryTransactions

```
neuraliumcli.dll QueryBlockBinaryTransactions <block-id> [--config=<str>]
    [--host=<str>] [--port=<int>] [--runtime-mode=<str>]
```

Query block's transaction in binary format (List<object>)

2.19.1 Parameters

1. BlockId (Required, Interger): the desired block's id

2.19.2 Return Value

The block's compressed bytes (list of json dictionaries)

2.20 QueryElectionContext

```
neuraliumcli.dll QueryElectionContext <block-id> [--config=<str>] [--host=<str>]
    [--port=<int>] [--runtime-mode=<str>]
```

Query a block's election context (object)

2.20.1 Parameters

1. BlockId (Required, Interger): the desired block's id

2.20.2 Return Value

The block's election context (json)

2.21 QueryWalletTransactionHistory

```
neuraliumcli.dll QueryWalletTransactionHistory <account-code> [--config=<str>]
    [--host=<str>] [--port=<int>] [--runtime-mode=<str>]
```

Query the wallet's transactions history (List<object>)

2.21.1 Parameters

1. AccountCode (Required, String): The account code

2.21.2 Return Value

The transaction history (list fo json dictionnaries).

2.22 QueryWalletAccounts

```
neuraliumcli.dll QueryWalletAccounts [--config=<str>] [--host=<str>] [--port=<int>]
    [--runtime-mode=<str>]
```

Query the wallet's accounts (List<object>)

2.22.1 Return Value

The wallet's accounts (list fo json dictionnaries). For example:

```
[
  {
    "accountCode": "XYZ",
    "accountId": "{*ABCD-ABCD-ABCD}",
    "friendlyName": "MyUserAccount",
    "isActive": true,
    "status": 1
  }
]
```

2.23 RenewLongRunningEvent

```
neuraliumcli.dll RenewLongRunningEvent <correlation-id> <key-correlation-code>
    <passphrase> [--config=<str>] [--host=<str>] [--port=<int>] [--runtime-mode=<str>]
```

Renew a long running event (bool)

2.23.1 Parameters

1. CorrelationId (Required, Unsigned Interger): The correlation id returned by the long running task that you started (e.g. *CreateNewWallet*)

2.23.2 Return Value

`true` if operation was sucessful

2.24 Shutdown

```
neuraliumcli.dll Shutdown [--config=<str>] [--host=<str>] [--port=<int>]
    [--runtime-mode=<str>]
```

Send a shutdown request to the server (bool)

2.24.1 Return Value

`true` if server was properly shutdown.

2.25 StartMining

```
neuraliumcli.dll StartMining <delegate-account-id> <tier> [--config=<str>]
    [--host=<str>] [--port=<int>] [--runtime-mode=<str>]
```

Starts the mining

2.25.1 Parameters

1. DelegateAccountId (Required, String): The account code on which mining shall be enabled
2. Tier (Required, Interget): the desired mining Tier (1,2,3 or 4)

2.25.2 Return Value

`true` if mining was enabled

2.26 StartMining

```
neuraliumcli.dll StopMining [--config=<str>] [--host=<str>] [--port=<int>]
    [--runtime-mode=<str>]
```

Stops the mining

2.26.1 Return Value

`true` if mining was disabled

2.27 SendNeuraliums

```
neuraliumcli.dll SendNeuraliums <target-account-id> <amount> <tp> <note>
    [--config=<str>] [--host=<str>] [--port=<int>] [--runtime-mode=<str>]
```

Send neuralium to another account [long-running] (uint)

2.27.1 Parameters

1. TargetAccountId (Required, String): The account code to send neuraliums to
2. Amount (Required, Decimal): The amount of iums to send (e.g. 10.42 *iums*)
3. Tip (Required, Decimal): The tip (in iums) to give to the miners for your trasaction to be prioritized

2.27.2 Return Value

The correlation id (Unsigned Interger)

2.28 WalletExists

```
neuraliumcli.dll WalletExists [--config=<str>] [--host=<str>] [--port=<int>]
    [--runtime-mode=<str>]
```

Can the wallet be found? (bool)

2.28.1 Return Value

`true` if a wallet can be found in your `.neuralium` folder.

2.29 WalletKeyFileCopied

Tells if wallet's Key file is finished copying

```
neuraliumcli.dll WalletKeyFileCopied <correlation-id> <key-correlation-code>
    [--config=<str>] [--host=<str>] [--port=<int>] [--runtime-mode=<str>]
```

2.29.1 Parameters

1. CorrelationId (Required, Unsigned Interger): The correlation id returned by the long running task that you started (e.g. *CreateNewWallet*)
2. KeyCorrelationId (Required, Unsigned Interger): The key correlation id returned by the long running task that you started (e.g. *CreateNewWallet*)

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search