SIChain Node Installation Guide for MacOS

1. Prerequisites

1. **Install Visual Studio Code**: This will simplify file management and command execution.
   - https://code.visualstudio.com/
2. **Open `SIChain_node` in Visual Studio Code**:
   - Open Visual Studio Code.
   - Create folder `~/SIChain_node`.
   - Use `File > Open Folder` to navigate to and open `~/SIChain_node`.
3. **Download the SIChain Node for macOS**:
   - Go to SIChain Node Download Page.
   - Download the **macOS version** of the SIChain node binary, extract it and move it to `~/SIChain_node`.
4. **Install Homebrew (if not already installed):**

- In terminal check if Homebrew is installed:

```
brew --version
```

- If you see `"command not found"`, install Homebrew using:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

- Follow the on-screen instructions.
- Verify the installation:

```
brew doctor
```

5. **Install `grpcurl` :**

```
brew install grpcurl
grpcurl --version
```

2. Configure `config.json`

1. In Visual Studio Code, open `config.json`.
2. Update the configuration to match the following (note that only `keys_file_path` and `secrets_path` should be adjusted):

```
{
  "client_api_endpoint": "0.0.0.0:9200",
  "master_node_client_api_endpoints": ["masternode.testnet.tolar.io:80"],
  "blockchain_explorer_client_api_endpoints": ["explorer.testnet.tolar.io:9876"],
  "key_store": {
    "keys_file_path": "/Users/$(whoami)/SIChain_node/keystore/keys.info",
    "secrets_path": "/Users/$(whoami)/SIChain_node/keystore/keys"
  },
  "max_grpc_threads": 100,
  "minimum_gas_price": "1000000000000",
  "network_id": 2
}
```

1. **Create the `keystore` Folder**:
   - Go to `~/SIChain_node`, create a new folder called `keystore`.

3. Start the Node

1. **Open a Terminal in Visual Studio Code**:
   - In Visual Studio Code, go to `View > Terminal` to open a terminal window.
2. **Start the Node**:
   - Grant execution permissions (if needed):

     ```
     chmod +x thin_node_bin
     ```

   - Run the following command in the terminal to start the node:

     ```
     ./thin_node_bin --config_path=/Users/$(whoami)/SIChain_node/config.json
     ```

3. **Verify the Node is Listening on Port 9200**:
   - **Check Listening Port**:
     - Open a new terminal tab in Visual Studio Code, or a separate terminal, and run:

       ```
       netstat -an | grep 9200
       ```

     - Confirm that port `9200` is in `LISTENING` status.

4. Initialize Keystore

1. **Create `create_keystore.json`**:

- In Visual Studio Code, create a file named `create_keystore.json` in `~/SIChain_node` with the following content:

  ```
  {
    "master_password": "your_master_password"
  }
  ```

2. **Run the Command to Create the Keystore**:

   - In the terminal, execute:

     ```
     grpcurl -plaintext -d @ 127.0.0.1:9200 tolar.proto.AccountService/Crea
     te < ~/SIChain_node/create_keystore.json
     ```

5. Open the Keystore

To open the keystore, use the following command:

```
grpcurl -plaintext -d @ 127.0.0.1:9200 tolar.proto.AccountService/Open < ~/SI
Chain_node/create_keystore.json
```

6. Create a New Address

1. **Create `create_address.json`**:

   - In Visual Studio Code, create a file named `create_address.json` in `~/SIChain_node` with this content:

     ```
     {
       "name": "TestnetAccount",
       "lock_password": "TestnetPassword",
       "lock_hint": "TestnetHint"
     }
     ```

2. **Generate a New Address**:

   - Run the following command to create a new address:

```
grpcurl -plaintext -d @ 127.0.0.1:9200 tolar.proto.AccountService/CreateNewAddress < ~/SIChain_node/create_address.json
```

7. List All Addresses

To list all addresses in the keystore, use:

```
grpcurl -plaintext -d "{}" 127.0.0.1:9200 tolar.proto.AccountService/ListAddresses
```

8. Decode Wallet Address

If the addresses are returned in Base64 format, you can use Base64 Decode to decode them to a human-readable hexadecimal format.

- **Create a file in `~/SIChain_node` named `wallet_address.txt` and save your decoded wallet address**.

9. Obtain Test TOL from Faucet

- Open the **SIChain Testnet Faucet**: https://tolar.io/testnet-faucet
- Copy your **decoded wallet address** from `wallet_address.txt`.
- Paste it into the faucet form and request tokens.
- Visit: Explorer
- Enter your **decoded wallet address** to check the latest transactions and balance.

10. Final Testing and Service Verification

Check Blockchain and Network Services

1. **Get Latest Block**:
    - Create a file `get_latest_block.json` in `~/SIChain_node` with `{}` as content.

- Run:

```
grpcurl -plaintext -d @ 127.0.0.1:9200 tolar.proto.BlockchainService/G
etLatestBlock < ~/SIChain_node/get_latest_block.json
```

```
{
  "blockIndex": "2667530",
  "hash": "MzIxNWZhZTQxOTVlM2JmODIwMmI3NDc1ZDM1OWM0ODI
yNWM1OTAxY2YwNjcwZTkyYzY1N2ZjZDk4Nzg1YjY0Zg==",
  "previousBlockHash": "Y2E0ZjBlZTAyZGU2ZmJkMjg4ZjI5OThhOThk
NTM0NjA3NzIwNmZlOThlNjE3YWZjMjczOWU0ODY1YzdkOWNhZg=
=",
  "confirmationTimestamp": "1724251098717"
}
```

2. **NetworkService Tests**:

- **Check Peer Count**:

```
grpcurl -plaintext -d "{}" 127.0.0.1:9200 tolar.proto.NetworkService/Pe
erCount
```

```
{
  "count": "1"
}
```

- **Check if Node is a Masternode**:

```
grpcurl -plaintext -d "{}" 127.0.0.1:9200 tolar.proto.NetworkService/Is
MasterNode
```

```
{

}
```

```
(An empty object if the node is not a masternode, or specific masterno
de information if it is.)
```

- **Get Number of Masternodes**:

```
grpcurl -plaintext -d "{}" 127.0.0.1:9200 tolar.proto.NetworkService/Ma
sterNodeCount
```

```
{
  "count": "1"
}
```

- **Get Maximum Peer Count**:

```
grpcurl -plaintext -d "{}" 127.0.0.1:9200 tolar.proto.NetworkService/Ma
xPeerCount
```

```
{
  "count": "42"
}
```

11. Supporting Resources

- **Official Documentation:**
  - **Tolar Official Website:** The central hub for all information related to Tolar HashNET, including news, documentations, updates, and links to official resources.
- **GitHub Repositories:**
  - **Tolar-HashNET GitHub:** The official GitHub organization for Tolar HashNET, hosting various repositories related to Tolar, including source code, libraries, and tools for developers.

    - **Tolar-HashNET Web3.js:** A JavaScript library for interacting with the Tolar blockchain, allowing developers to build decentralized applications (dApps) that communicate with the Tolar network.
    - **Tolar-HashNET Web3j:** A Java library for integrating Java-based applications with the Tolar blockchain, providing tools for interacting with smart contracts and blockchain data.
- **Web Wallet and Tolar Scanner:**
    - **Tolar Testnet Faucet:** A tool for developers to obtain test TOL tokens for use on the Tolar testnet, useful for testing transactions and smart contracts without using real funds.
    - **Tolar Explorer (Testnet):** An online blockchain explorer for viewing and monitoring transactions, blocks, and accounts on the Tolar testnet.
- **gRPC API Documentation:**
    - **gRPC API Documentation:** Detailed documentation on the available gRPC API endpoints for interacting with the Hashnet blockchain. This documentation is essential for developers implementing blockchain interactions such as account management, transactions, and network operations.