10.0 Testing

10.1 Unit Test Plan

The purpose of this section is to test basic node methods. This is to guarantee nodes do not fail when performing local functionality.

10.1.1 Unit Test Descriptions

10.1.1.1 Unit Test 1 Given a set of valid transaction test generate block

10.1.1.2 Unit Test 2 Given a set including some invalid transaction test generate block only includes the valid transactions

10.1.1.3 Unit Test 3 Given a valid generated block and its transactions check the validity of the new block

10.1.1.4 Unit Test 4 Given a invalid generated block and its transactions check the block is invalid

10.1.1.5 Unit Test 5 Given an unsigned json wrapper and private key guarantee the wrapper is correctly signed

10.1.1.6 Unit Test 6 Given a correctly signed json wrapper and verify the wrapper is correctly signed

10.1.1.7 Unit Test 7 Given an incorrectly signed json wrapper and verify the wrapper is incorrectly signed

10.1.1.8 Unit Test 8 Given a public key generate the account number and check its validity

10.1.1.9 Unit Test 9 Verify account creation returns a valid private key

10.2 Integration Test Plan

The debased system uses a Puppet server in order to perform integration tests. Puppet is a modern software configuration management system supported by the open source community. The reason for using Puppet is due to how flexible and open ended Puppet is. After installing Puppet and setting up the server, Puppet can be used to simulate a decentralized peer-to-peer system that powers debased itself. From here, various commands can be executed and the results can be used and checked by Puppet.

10.2.1 Setting up a simple debased system

Each test here must occur in order listed.

10.2.1.1 Spawn the first node

10.2.1.2 Launch the second node

10.2.1.3 Create a stream between the first and the second node

10.2.1.4 First node sends a validated transaction to the second node

10.2.1.5 Second node validates the transaction from the first node

10.2.1.6 Second node records the balance change and query in its block

10.2.1.7 Terminate the second node safely

10.2.1.8 Ensure the first node is still running properly

10.2.2 Setting up a complex debased system

Each test here must occur in order presented.

10.2.2.1 Spawn first and second node
10.2.2.2 Create a stream between the first and second node
10.2.2.3 Spawn the third and fourth node
10.2.2.4 Create a stream between the first and third node
10.2.2.5 Create a stream between the second and fourth node
10.2.2.6 First node sends a validated transaction to the second node
10.2.2.7 Second node validates the transaction from the first node
10.2.2.8 Second node records the balance change and query in its block
10.2.2.9 Third node sends a validated transaction to the fourth node
10.2.2.10 Fourth node validates the transaction from the first node
10.2.2.11 Fourth node records the balance change and query in its block
10.2.2.12 Terminate the third node safely
10.2.2.13 Ensure the first, second, and fourth is still running properly

10.2.2 Setting up a complex debased system

10.3 Module Dependencies