deep dive technology

ABBC Gen.2 Exchange Instructions

Services Performed for:

ABBC FOUNDATION

Boris Heismann
Chief Information Officer

E: boris.heismann@deepdive.tech
W: www.deepdive.tech

Date: July 20, 2019 Revision: v1.01

Exchange Instructions

TABLE OF CONTENTS

STATEMENT OF CONFIDENTIALITY

This document has been prepared specifically for ABBC Foundation (ABBC). Deep Dive Technology Inc. (Deep Dive Technology) requests that its contents not be disclosed to third parties without the prior written consent of Deep Dive Technology.

DOCUMENT REVISION HISTORY

Any changes, additions, or deletions can be forwarded to the document author.

Note: Version Release Updates are as Follows:

1. Major Changes in Procedures or Wording

.01 Minor Changes and Document Edits

Revision	Date	Description of Revisions	Revised By
0.5	Jul 01, 2019	Document skeleton	Boris Heismann
0.9	Jul 15, 2019	Document draft review	Boris Heismann & Misha Hanin
1.0	Jul 20, 2019	Document draft review	Boris Heismann
1.01	Jul 23, 2019	Final review	Tad Einstein
1.02	Aug. 7, 2019	Update to technical details	Boris Heismann
1.03	Feb. 18, 2020	Review / update	Illia Bondarchuk

EXECUTIVE SUMMARY

ABBC coin is migrating to a new blockchain technology which will allow for faster transaction times, zero fees when staking system resources, with more advanced features to come in the future.

We will call the existing **ABBC Blockchain Gen.1** and the new blockchain -- **ABBC Blockchain Gen.2**.

MIGRATION

Every current ABBC holder, at time of the cutover, will transfer their ABBC balance from the **Gen.1** to the **Gen.2** Blockchain.

For **Aladdin** and **MC Wallet** users, there will be an automated migration flow.

For large balances, we will provide a **manual migration flow**.

ABBC GEN.2 BLOCKCHAIN DETAILS

ABBC Gen.2 Blockchain is based on **EOS**. You can read more about the EOS blockchain here:

https://developers.eos.io/

If you have already added support for **EOS** or **EOS tokens** on your exchange then it will be **very easy** to do the same to support **ABBC Gen.2**.

PROCESSING DEPOSITS

Overview

It is recommended to build **ABBC Gen.2** deposit processing on the same architecture as **EOS**. Here is how it will work:

- Exchange sets up a read-only node in the ABBC Gen.2 blockchain, configures one of the history plug-ins on this node (state-history-plugin is standard, but there are other plug-ins for monitoring blockchain state, see more here: https://developers.eos.io/manuals/eos/latest/nodeos/plugins/state_history_plugin/index)
- Then a single deposit account is created on the ABBC Gen.2 blockchain. Unlike Bitcoin, Ethereum, and all other cryptocurrencies, it is **recommended** to have a **single deposit account** where users are **distinguished** by **transaction memo**.

- For each exchange user a unique memo is generated. Later when a transaction to the deposit address is detected, the exchange will know which user deposited funds by checking the transaction memo field.

As mentioned, tracking deposits works similar to EOS, you can track transactions from the same system token contract (**eosio.token**), but a different currency (ABBC, precision **8** instead of standard EOS precision **4**).

Another difference between EOS and ABBC is that there is an extra action to track on ABBC Gen 2. All EOS transfers are triggered by the ***transfer*** action. For ABBC, you have to monitor the ***transferx*** action additionally (*which is the way that ABBC Gen.2 users are able to send funds without staking resources*).

Details

To get started you will need to do the following:

- Create a deposit account on the ABBC Blockchain. Since only existing account holders can create new account please contact **business@abbcfoundation.com** to get started
- Set up ABBC node. Use EOS node setup instructions from here:

  https://developers.eos.io/manuals/eos/latest/nodeos/usage/index

  and here

  https://developers.eos.io/manuals/eos/latest/nodeos/plugins/state_history_plugin/index

  When connecting to the blockchain use the following P2P peers:

  1.p2p.node.abbcnet.io:9010
  2.p2p.node.abbcnet.io:9010
  3.p2p.node.abbcnet.io:9010
  4.p2p.node.abbcnet.io:9010
  5.p2p.node.abbcnet.io:9010
  6.p2p.node.abbcnet.io:9010
  7.p2p.node.abbcnet.io:9010

  You should also use the **genesis.json** file to start syncing nodes. You can receive this file by contacting:

  business@abbcfoundation.com

- Wait until the node is synced, and then test incoming transactions to see how they appear in the state history database. Then hook this up together with user-unique memos for deposits to your exchange's deposit system.

PROCESSING WITHDRAWALS

Overview

User withdrawals are simple transfers to user accounts. Basically, a user enters an account name to withdraw funds to and the exchange triggers the transfer from the hot wallet.

To prepare serializing and signing transactions, you can use standard EOS libraries available for multiple server environments. Here are some examples below:

- JavaScript: https://github.com/EOSIO/eosjs
- Python: https://github.com/eosmoto/eosiopy
- Java: https://github.com/EOSIO/eosio-java

When creating transactions, you will be calling the same transfer method of the **eosio.token** contract as in the EOS blockchain, but you must specify a precision amount of 8 for the ABBC currency (*for example, 10 ABBC transfer will look like this 10.00000000 ABBC as transaction parameter*).

Signed transactions can be pushed to the blockchain using the same node as used for processing user deposits.

Exchanges might choose to support ABBC-specific transfers to user's public keys. This is an ABBC-only feature: users do not have to create accounts to receive funds, instead transfers are done to a user's public key and once the blockchain records the transfer it will create a new account for the user for free. To support such withdrawals exchanges have to allow users to enter their public key (or ABBC address) instead of the account name and use the ***transferpub*** method instead of ***transfer***. Otherwise the process is the same as with a regular transfer.

Details

Before you can start issuing ABBC withdrawals you will need an ABBC account for the hot wallet, some ABBC tokens on it, and access to an ABBC blockchain node.

You can use the IP address of your own node created in the **deposits processing step**, but you can also use our nodes with the following hostname and port:

https://1.rpc.node.abbcnet.io
https://2.rpc.node.abbcnet.io
https://3.rpc.node.abbcnet.io

Since **no private data gets stored on the nodes** you can **safely use any node available to you**.

Then, to create a new transaction, use the following code snippet (*this example is for EOS JavaScript library*):

```
const result = await api.transact({
    actions: [{
      account: 'eosio.token',
      name: 'transfer',
      authorization: [{
        actor: 'hotwalletacc',
        permission: 'active',
      }],
      data: {
        from: 'hotwalletacc',
        to: 'destnuseracc',
        quantity: '1.00000000 ABBC',
        memo: '',
      },
    }]
  },
    blocksBehind: 3,
    expireSeconds: 30,
  });
```

Where ***hotwalletacc*** is your account address (sender) and ***destuseracc*** is a recipient of the transaction.

You can find more details on how to initialize **EOSJS** and push transactions to the blockchain here: https://github.com/EOSIO/eosjs