Stake Usage Guide

USER MANUAL

This manual was developed by GolosCore team for CyberWay blockchain users and can be used as reference material. The manual contains a brief description of the operations performed on the staked tokens, as well as examples with their use.

Terms used

Delegator — a user who delegates part of his/her stake to another user.

Stake — bandwidth share (RAM, NET, CPU and Storage) allocated to a user in the system for executing transactions. These resources are presented at their cost in the form of tokens. The user can dispose the share of resources both independently and/or hand it over to another user (delegate the share of resources).

Staked tokens — tokens allocated for the use of the part a part of the system resources. A staked tokenrepresents a token that has been locked up for a time period.

The concept of using system resources

User activity within the network (publish posts, send comments, vote, execute token operations, etc.) such as system resources are available to him - bandwidth resources (CPU, NET, RAM and Storage). A more active user requires more system resources. Amount of system resources available to the user depends on the number of the user's staked tokens.

A stake is a cost (in tokens) of system resources. For example, if the stake is 1000 tokens, of which 50 tokens belong to a user, then this user is allocated a share of the system resources (CPU, NET, RAM and Storage) in the amount of 5 %.

To enable a user to execute transactions, the user has to transfer part of the active tokens to stake. User activity is limited by the number of staked tokens. The user can increase the share of bandwidth. To do this, she/he should acquire active tokens and transfer them to the stake (get additional resources).

If the user's activity decreases, she/he can reduce the share of the system’s resources. To do this, she/he has to either withdraw a part of the tokens back from the stake to the active state, or delegate this part to another user.

When executing transactions, a user does not have to worry about which specific resource will be consumed more (or less) and how the staked tokens should be spent. The system dynamically and optimally distributes the user's staked tokens.

Stake token operations

transfer

`transfer` operation is intended to transfer active tokens to the stake. This operation is executed by the `cyber.token` contract. `cyber.stake` contract should be specified as a recipient of the funds.

Option_1. Transfer tokens to a stake for yourself.

```
cleos push action cyber.token transfer ‘[<user account>, cyber.stake, “qu
```

Arguments:

- `user account` — user transferring tokens to a stake.
- `quantity CYBER` — number of tokens being transferred.
- `active key` — user’s active key.

The liquid balance of user tokens in `cyber.token` contract is reduced by the `quantity`. The balance of the user's staked tokens in `cyber.stake` contract is increased by the `quantity`.

Example:

```
cleos push action cyber.token transfer '[alice, cyber.stake, "100.0000 CY
```

User `alice` transfers 100 CYBER tokens to stake, due to which she can increase her activity on the network. The command is signed with the active key `alice@active`.

Option_2. Token transfer to another user's stake.

```
cleos push action cyber.token transfer '[<user account>, cyber.stake, "qu
```

The `user account` argument is a recipient of the staked tokens.

Example:

```
cleos push action cyber.token transfer '[alice, cyber.stake, "100.0000 CY
```

User `alice` transfers 100 CYBER tokens to stake for user `bob`. In this case, the `alice` liquid balance will be reduced by the 100 CYBER. The `bob` balance in `cyber.stake` contract is increased by 100 staked tokens. The command is signed with the active key `alice@active`.

Option_3. Transfer tokens to a stake via using the "system stake" operation.

```
cleos system stake <user account> "quantity CYBER"
```

Example_1: Transfer tokens to a stake for yourself.

```
cleos system stake alice "100.0000 CYBER"
```

User `alice` transfers 100 CYBER tokens to stake. Unlike `transfer`, this operation does not require an active key signature.

Example_2: Transfer tokens to a stake for another user.

```
cleos system stake alice “100.0000 CYBER” --beneficiary bob
```

User `alice` transfers 100 CYBER tokens to stake for user `bob`. Unlike `transfer`, this operation does not require an active key signature.

withdraw

`withdraw` operation is intended to withdraw tokens from the stake to active state. This operation is executed by the smart contract `cyber.stake`.

```
cleos push action cyber.stake withdraw <account name> “quantity CYBER”
```

Tokens are withdrawn immediately without any delay. The withdrawal is executed if the remainder of user’s stake, taking into account the funds being withdrawn, as well as the funds delegated to other users, covers the costs of the resources used by the user.

Example:

```
cleos push action cyber.stake withdraw alice “100.0000 CYBER”
```

After this operation is completed, user `alice` stake will decrease by 100 CYBER tokens. At the same time, active tokens will be credited to the `alice` account balance.

delegateuse

`delegateuse` operation is intended to transfer part of the stake resources (RAM, NET, CPU, Storage) to another user. However, resources are not directly delegated. Instead of resources, their cost is delegated — number of staked tokens.

The operation is executed by the smart contract `cyber.stake`.

```
cleos push action cyber.stake delegateuse '[<delegator account>, <recipie
```

Arguments:

- `delegator account` — user delegating staked tokens.
- `quantity CYBER` — number of staked tokens.
- `active key` — delegator's active key.

Example_1:

```
cleos push action cyber.stake delegateuse '[alice, bob, "10.0000 CYBER"] -
```

`Alice` user delegates 10 tokens to user `bob`. The operation is signed by the `alice` active key.

Example_2: `delegatebw` operation can be executed using the specialized cleos command.

```
cleos system delegatebw alice bob "10.0000 CYBER"
```

undelegatebw

`undelegatebw` operation is intended to return delegated stake. This operation is performed in two stages:

- **Stage_1** request for a return of delegated staked tokens.
- **Stage_2** crediting returned amount of staked tokens to a stake.

Stage_1 `recalluse` operation can be executed to revoke a delegated stake:

```
cleos push action cyber.stake recalluse '[<delegator account>, <delegated
```

Example_1:

```
cleos push action cyber.stake recalluse '[alice, bob, "10.0000 CYBER"] -p
```

User `alice` revokes 10 tokens that were delegated to user `bob.` The operation is signed by the `alice` active key.

Example_2: The operation `undelegatebw` can also be executed using the specialized cleos command

```
cleos system undelegatebw alice bob "10.0000 CYBER"
```

Stage_2 `claim` operation can be executed to credit the returned amount of staked tokens to a stake:

```
cleos push action cyber.stake claim '[alice, bob, "CYBER"] -p  alice@acti
```

or execute the specialized cleos command

```
cleos system claimbw alice bob "CYBER"
```

Stage_2 operations should only be performed 30 days after completion of stage_1 operations.

> **Note:**
> The differences between delegation and crediting to another user are:

> - when delegated, the staked tokens are transferred. These tokens can be taken back without the recipient's signature.
> - when crediting to a stake, the transfer comes from liquid balance.

newaccount

`newaccount` operation is intended to create a user and delegate staked tokens to him/her.

```
cleos system newaccount <creator account> <new account> <quantity of stak
```

If the `--transfer` flag is added to this command line, then staked tokens will be transferred irrevocably to the created account.

listbw

`listbw` operation is intended to obtain a list of users to whom a stake has been delegated

```
cleos system listbw <delegator account>
```

The delegator receives a list of users to whom he/she has delegated the stake.