Creating Wallet and Keys for Development

Creating Wallet and Keys for Development

User Guide

This section provides instructions for creating a wallet for use in developing or maintaining software. Before the user can make a change to the program code of the product, he needs to create a wallet and keys for development.

Wallet — a storage for a pair of keys — public private ones. The private key is stored in encrypted form and is used to sign transactions. Access to the wallet is provided from the command line using the `cleos` application.

It's important

> Tokens are not stored in the wallet. The wallet stores only the keys for signing transactions. The user creates a transaction and sends it to the wallet for signature. The wallet returns a signed transaction, after which it is transferred to the network. If it is confirmed that the transaction is valid and contains authenthic signatures, it is included in the chain block.

1 Creating a wallet

To create a wallet, you have to perform the `create` operation.

```
$ cleos wallet create --to-console
```

The `--to-console` option sets the default operation with the session output to the console. To use the `cleos` application, not for the development process, use the `--to-file` option to prevent the wallet password from entering the bash history. Since keys are created for development and not for use in a functioning Mainnet product, setting the `--to-console` option does not threaten security.

The cleos application tells you the password to save. Further information appears with a reminder that this password will be needed when unlocking the wallet, and that without a password, it will be impossible to recover the imported keys.

```
Creating wallet: default
Save password to use in the future to unlock this wallet.
Without password imported keys will not be retrievable."PW5Kewn9L76X8Fpd...
```

Running the default command creates a wallet named «default». If you want to create a wallet with a different name (for example, when creating more than one wallet) you can use the `--name` option (or `-n`).

```
$ cleos wallet create --name second-wallet --to-console
```

As a result, a wallet with the `second-wallet` name is created.

2 Opening a wallet

Wallets are stored in the `keosd` application. Wallets are kept in the closed state by default. To open the wallet, you have to use the `open` operation.

```
$ cleos wallet open
or
$ cleos wallet open --name second-wallet
```

The following information should appear:

```
Opened: default
or
Opened: second-wallet
```

To obtain a list of open wallets use the `list` operation.

```
$ cleos wallet list
```

As a result, the following information should appear:

```
Wallets:
[
  "default",
  "second-wallet"
]
```

In the absence of purses in the open state, the following information will appear:

```
Wallets:
[
]
```

3 Unlocking a wallet

Despite the fact that the wallet is open, the `keosd` application keeps it in a locked state. To use the wallet, it must be unlocked by `unlock` operation.

```
$ cleos wallet unlock
or
$ cleos wallet unlock --name second-wallet
```

A password prompt will appear. You may enter the password and press «enter».

Please note:

> You can enter the password directly on the command line by adding the `--password` option. For example,

```
$ cleos wallet unlock --password PW5...w2
```

You can get a list of open wallets by re-executing:

```
$ cleos wallet unlock
or
$ cleos wallet unlock --name second-wallet
```

The following information should appear:

```
Wallets:
[
  "default *",
  "second-wallet *"
]
```

The presence of the «*» symbol means that the wallet is in the unlocked state.

4 Downloading a key into the wallet created

After the wallet is created and unlocked, a pair of keys can be loaded into it - private and public. To do this, you can use the `create_key` operation. This operation allows you to generate keys and automatically load them into the wallet. By default, a key with type «K1» — privileged will be generated.

```
$ cleos wallet create_key
```

When having more than one wallet, you have specify the name of the wallet in the team, adding the `-name <text>` option.

```
$ cleos wallet create_key  --name second-wallet
```

The following message on the creation of a pair of private and public keys should appear.

```
Created new private key with a public key of: "GLS8PE...,X6P..."
```

Please note:

> Unlike EOS, in CyberWay the public key code actually starts with the «GLS» characters.

5 Importing the development key

In order to incorporate the key into development you have to perform the `import` operation.

```
$ cleos wallet import
```

Next, you need to enter the proposed system code for the development key. Currently, the following code is used:

```
5KQwrPbwdL6PhXujxW37FSSQZ1JiwsST4cqQzDeyXtP79zkvFD3
```

> *Do not use the key to develop a program for any other purposes. This is predominantly associated with a high risk of losing access to a user account forever. The private key development is not protected.*

$ cleos wallet lock_all

6 Locking a wallet (wallets)

Sometimes it's necessary to have your wallet locked. For instance, when the long-term interruptions in software development are occuring. To lock a single wallet you can use the lock operation.

```
$ cleos wallet lock
or
$ cleos wallet lock --name second-wallet
```

A message about locking the user's wallet should appear.

```
Locked: 'default'
or
Locked: 'second-wallet'
```

To block all user wallets, use the `lock_all` operation.

```
$ cleos wallet lock_all
```

The following message on locking all user wallets should appear.

```
Locked All Wallets
```